

OrCAD PSpice®

User's Guide

Copyright © 1998 OrCAD, Inc. All rights reserved.

Trademarks

OrCAD, OrCAD Layout, OrCAD Express, OrCAD Capture, OrCAD PSpice, and OrCAD PSpice A/D are registered trademarks of OrCAD, Inc. OrCAD Capture CIS, and OrCAD Express CIS are trademarks of OrCAD, Inc.

Microsoft, Visual Basic, Windows, Windows NT, and other names of Microsoft products referenced herein are trademarks or registered trademarks of Microsoft Corporation.

All other brand and product names mentioned herein are used for identification purposes only, and are trademarks or registered trademarks of their respective holders.

Part Number 60-30-636

First edition 30 November 1998

Technical Support	(503) 671-9400
Corporate offices	(503) 671-9500
OrCAD Japan K.K.	81-45-621-1911
OrCAD UK Ltd.	44-1256-381-400
Fax	(503) 671-9501
General email	info@orcad.com
Technical Support email	techsupport@orcad.com
World Wide Web	http://www.orcad.com
OrCAD Design Network (ODN)	http://www.orcad.com/odn



9300 SW Nimbus Ave.
Beaverton, OR 97008 USA

Contents

Before you begin xxiii

Welcome to OrCAD	xxiii
OrCAD PSpice overview	xxiv
How to use this guide	xxv
Typographical conventions	xxv
Related documentation	xxvi
Online Help	xxvii
If you have the demo CD-ROM	xxviii
OrCAD demo CD-ROM	xxviii
What's New	xxix

Part one Simulation primer

Chapter 1 Things you need to know 1

Chapter overview	1
What is PSpice?	2
Analyses you can run with PSpice	3
Basic analyses	3
DC sweep & other DC calculations	3
AC sweep and noise	4
Transient and Fourier	5
Advanced multi-run analyses	6
Parametric and temperature	6
Monte Carlo and sensitivity/worst-case	7
Analyzing waveforms with PSpice	8
What is waveform analysis?	8
Using PSpice with other OrCAD programs	9
Using Capture to prepare for simulation	9
What is the Stimulus Editor?	9
What is the Model Editor?	10

Files needed for simulation	10
Files that Capture generates	10
Netlist file	11
Circuit file	11
Other files that you can configure for simulation	11
Model library	12
Stimulus file	13
Include file	13
Configuring model library, stimulus, and include files	13
Files that PSpice generates	14
Waveform data file	14
PSpice output file	14

Chapter 2 Simulation examples 15

Chapter overview	15
Example circuit creation	16
Finding out more about setting up your design	21
Running PSpice	22
Performing a bias point analysis	22
Using the simulation output file	24
Finding out more about bias point calculations	25
DC sweep analysis	26
Setting up and running a DC sweep analysis	26
Displaying DC analysis results	28
Finding out more about DC sweep analysis	31
Transient analysis	32
Finding out more about transient analysis	36
AC sweep analysis	37
Setting up and running an AC sweep analysis	37
AC sweep analysis results	39
Finding out more about AC sweep and noise analysis	41
Parametric analysis	42
Setting up and running the parametric analysis	43
Analyzing waveform families	45
Finding out more about parametric analysis	48
Performance analysis	49
Finding out more about performance analysis	51

Part two Design entry

Chapter 3 Preparing a design for simulation 55

Chapter overview	55
Checklist for simulation setup	56
Typical simulation setup steps	56
Advanced design entry and simulation setup steps	57
When netlisting fails or the simulation does not start	58
Things to check in your design	58
Things to check in your system configuration	59
Using parts that you can simulate	60
Vendor-supplied parts	61
Part naming conventions	61
Finding the part that you want	62
Passive parts	64
Breakout parts	65
Behavioral parts	66
Using global parameters and expressions for values	67
Global parameters	67
Declaring and using a global parameter	67
Expressions	69
Specifying expressions	69
Defining power supplies	74
For the analog portion of your circuit	74
Defining stimuli	75
Analog stimuli	75
Using VSTIM and ISTIM	76
If you want to specify multiple stimulus types	77
Things to watch for	79
Unmodeled parts	79
Do this if the part in question is from the OrCAD libraries	79
Check for this if the part in question is custom-built	81
Unconfigured model, stimulus, or include files	81
Check for this	82
Unmodeled pins	82
Check for this	83
Missing ground	83
Check for this	83
Missing DC path to ground	84
Check for this	84

Chapter 4 Creating and editing models 85

Chapter overview	85
What are models?	87
Models defined as model parameter sets	87
Models defined as subcircuit netlists	87
How are models organized?	88
Model libraries	88
Model library configuration	89
Global vs. design models and libraries	89
Nested model libraries	90
OrCAD-provided models	90
Tools to create and edit models	91
Ways to create and edit models	92
Using the Model Editor to	
edit models	93
Ways to use the Model Editor	94
Model Editor-supported device types	95
Ways To Characterize Models	96
Creating models from data sheet information	96
Analyzing the effect of model parameters	
on device characteristics	97
How to fit models	97
Running the Model Editor alone	99
Starting the Model Editor	99
Enabling and disabling automatic part creation	100
Saving global models (and parts)	100
Running the Model Editor from the schematic page editor	101
What is an instance model?	101
Starting the Model Editor	102
Saving design models	102
What happens if you don't save the instance model	103
The Model Editor tutorial	104
Creating the half-wave rectifier design	104
Using the Model Editor to edit the D1 diode model	105
Entering data sheet information	105
Extracting model parameters	107
Adding curves for more than one temperature	108
Completing the model definition	109
Editing model text	110
Editing .MODEL definitions	110
Editing .SUBCKT definitions	111

Changing the model name	111
Starting the Model Editor	
from the schematic page editor in Capture	111
What is an instance model?	112
Starting the Model Editor	112
Saving design models	113
Example: editing a Q2N2222 instance model	114
Starting the Model Editor	114
Editing the Q2N2222-X model instance	114
Saving the edits and updating the schematic	115
Using the Create Subcircuit command	115
Changing the model reference to an existing model definition	117
Reusing instance models	118
Reusing instance models in the same schematic	118
Making instance models available to all designs	119
Configuring model libraries	120
The Libraries and Include Files tabs	120
How PSpice uses model libraries	121
Search order	121
Handling duplicate model names	122
Adding model libraries to the configuration	122
Changing design and global scope	123
Changing model library search order	124
Changing the library search path	125

Chapter 5 Creating parts for models 127

Chapter overview	127
What's different about parts used for simulation?	129
Ways to create parts	
for models	129
Preparing your models for part creation	130
Using the Model Editor to create parts	131
Starting the Model Editor	131
Setting up automatic part creation	132
Basing new parts on a custom set of parts	133
Editing part graphics	135
How Capture places parts	135
Defining grid spacing	136
Grid spacing for graphics	136
Grid spacing for pins	136
Attaching models to parts	138

MODEL	138
Defining part properties needed for simulation	139
PSPICETEMPLATE	140
PSPICETEMPLATE syntax	140
PSPICETEMPLATE examples	143

Chapter 6 Analog behavioral modeling 147

Chapter overview	147
Overview of analog behavioral modeling	148
The ABM.OLB part library file	149
Placing and specifying ABM parts	150
Net names and device names in ABM expressions	150
Forcing the use of a global definition	151
ABM part templates	152
Control system parts	153
Basic components	155
Limiters	156
Chebyshev filters	157
Integrator and differentiator	160
Table look-up parts	160
Laplace transform part	164
Math functions	167
ABM expression parts	168
An instantaneous device example: modeling a triode	171
PSpice-equivalent parts	174
Implementation of PSpice -equivalent parts	175
Modeling mathematical or instantaneous relationships	176
EVALUE and GVALUE parts	176
EMULT, GMULT, ESUM, and GSUM	178
Lookup tables (ETABLE and GTABLE)	179
Frequency-domain device models	181
Laplace transforms (LAPLACE)	181
Frequency response tables (EFREQ and GFREQ)	183
Cautions and recommendations for simulation and analysis	186
Instantaneous device modeling	186
Frequency-domain parts	187
Laplace transforms	187
Non-causality and Laplace transforms	188
Chebyshev filters	190
Frequency tables	190
Trading off computer resources for accuracy	191

Basic controlled sources	192
Creating custom ABM parts	192

Part three Setting Up and Running Analyses

Chapter 7 Setting up analyses and starting simulation 195

Chapter overview	195
Analysis types	196
Setting up analyses	197
Execution order for standard analyses	198
Output variables	199
Modifiers	200
Starting a simulation	206
Starting a simulation from Capture	206
Starting a simulation outside of Capture	207
Setting up batch simulations	207
Multiple simulation setups within one circuit file	207
Running simulations with multiple circuit files	208
The PSpice simulation window	208

Chapter 8 DC analyses 213

Chapter overview	213
DC Sweep	214
Minimum requirements to run a DC sweep analysis	214
Overview of DC sweep	216
Setting up a DC stimulus	218
Nested DC sweeps	219
Curve families for DC sweeps	221
Bias point	223
Minimum requirements to run a bias point analysis	223
Overview of bias point	223
Small-signal DC transfer	225
Minimum requirements to run a small-signal DC transfer analysis	225
Overview of small-signal DC transfer	226
DC sensitivity	228
Minimum requirements to run a DC sensitivity analysis	228
Overview of DC sensitivity	229

Chapter 9 AC analyses 231

Chapter overview	231
----------------------------	-----

AC sweep analysis	232
Setting up and running an AC sweep	232
What is AC sweep?	232
Setting up an AC stimulus	233
Setting up an AC analysis	235
AC sweep setup in example.opj	237
How PSpice treats nonlinear devices	239
What's required to transform a device into a linear circuit	239
What PSpice does	239
Example: nonlinear behavioral modeling block	239
Noise analysis	241
Setting up and running a noise analysis	241
What is noise analysis?	242
How PSpice calculates total output and input noise	242
Setting up a noise analysis	243
Analyzing Noise in the Probe window	245
About noise units	246
Example	246

Chapter 10 Transient analysis 249

Chapter overview	249
Overview of transient analysis	250
Minimum requirements to run a transient analysis	250
Minimum circuit design requirements	250
Minimum program setup requirements	250
Defining a time-based stimulus	252
Overview of stimulus generation	252
The Stimulus Editor utility	253
Stimulus files	253
Configuring stimulus files	254
Starting the Stimulus Editor	254
Defining stimuli	256
Example: piecewise linear stimulus	256
Example: sine wave sweep	257
Creating new stimulus symbols	259
Editing a stimulus	260
To edit an existing stimulus	260
To edit a PWL stimulus	260
To select a time and value scale factor for PWL stimuli	260
Deleting and removing traces	261

Manual stimulus configuration	261
To manually configure a stimulus	261
Transient (time) response	263
Internal time steps in transient analyses	265
Switching circuits in transient analyses	266
Plotting hysteresis curves	266
Fourier components	268
Chapter 11 Parametric and temperature analysis 271	
Chapter overview	271
Parametric analysis	272
Minimum requirements to run a parametric analysis	272
Overview of parametric analysis	273
RLC filter example	274
Entering the design	274
Running the simulation	275
Using performance analysis to plot overshoot and rise time	275
Example: frequency response vs. arbitrary parameter	278
Setting up the circuit	278
Temperature analysis	281
Minimum requirements to run a temperature analysis	281
Overview of temperature analysis	282
Chapter 12 Monte Carlo and sensitivity/worst-case analyses 283	
Chapter overview	283
Statistical analyses	284
Overview of statistical analyses	284
Output control for statistical analyses	285
Model parameter values reports	285
Waveform reports	286
Collating functions	287
Temperature considerations in statistical analyses	288
Monte Carlo analysis	289
Reading the summary report	291
Example: Monte Carlo analysis of a pressure sensor	293
Drawing the schematic	293
Defining part values	294
Setting up the parameters	295
Using resistors with models	296
Saving the design	297
Defining tolerances for the resistor models	297

Setting up the analyses	299
Running the analysis and viewing the results	300
Monte Carlo Histograms	301
Chebyshev filter example	301
Creating models for Monte Carlo analysis	302
Setting up the analysis	302
Creating histograms	303
Worst-case analysis	306
Overview of worst-case analysis	306
Inputs	307
Procedure	307
Outputs	308
Caution: An important condition for correct worst-case analysis	308
Worst-case analysis example	309
Tips and other useful information	313
VARY BOTH, VARY DEV, and VARY LOT	313
Gaussian distributions	314
YMAX collating function	314
RELTOL	314
Sensitivity analysis	314
Manual optimization	314
Monte Carlo analysis	315

Part four Viewing results

Chapter 13 Analyzing waveforms 319

Chapter overview	319
Overview of waveform analysis	320
Elements of a plot	321
Elements of a Probe window	322
Managing multiple Probe windows	323
Printing multiple windows	323
Setting up waveform analysis	324
Setting up colors	324
Editing display and print colors in the PSPICE.INI file	324
Configuring trace color schemes	326
Viewing waveforms	327
Setting up waveform display from Capture	327
Viewing waveforms while simulating	328
Configuring update intervals	329

Interacting with waveform analysis during simulation	329
Pausing a simulation and viewing waveforms	330
Using schematic page markers to add traces	331
Limiting waveform data file size	334
Limiting file size using markers	334
Limiting file size by excluding internal subcircuit data	336
Limiting file size by suppressing the first part of simulation output	336
Using simulation data from multiple files	337
Appending waveform data files	337
Adding traces from specific loaded waveform data files	338
Saving simulation results in ASCII format	339
Analog example	341
Running the simulation	341
Displaying voltages on nets	343
User interface features for waveform analysis	344
Zoom regions	344
Scrolling traces	346
Modifying trace expressions and labels	346
Moving and copying trace names and expressions	347
Copying and moving labels	348
Tabulating trace data values	349
Using cursors	350
Displaying cursors	350
Moving cursors	351
Example: using cursors	352
Tracking simulation messages	354
Message tracking from the message summary	354
The Simulation Message Summary dialog box	354
Persistent hazards	355
Message tracking from the waveform	356
Trace expressions	356
Basic output variable form	357
Output variable form for device terminals	358
Analog trace expressions	364
Trace expression aliases	364
Arithmetic functions	364
Rules for numeric values suffixes	365

Chapter 14 Other output options 367

Chapter overview	367
----------------------------	-----

Viewing analog results in the PSpice window	368
Writing additional results to the PSpice output file	369
Generating plots of voltage and current values	369
Generating tables of voltage and current values	370

Appendix A Setting initial state 373

Appendix overview	373
Save and load bias point	374
Save bias point	374
Load bias point	375
Setpoints	376
Setting initial conditions	378

Appendix B Convergence and “time step too small errors” 379

Appendix overview	379
Introduction	380
Newton-Raphson requirements	380
Is there a solution?	381
Are the Equations Continuous?	382
Are the derivatives correct?	382
Is the initial approximation close enough?	383
Bias point and DC sweep	385
Semiconductors	385
Switches	386
Behavioral modeling expressions	387
Transient analysis	388
Skipping the bias point	389
The dynamic range of TIME	389
Failure at the first time step	390
Parasitic capacitances	391
Inductors and transformers	391
Bipolar transistors substrate junction	392
Diagnostics	393

Index 395

Figures

Figure 1	User-configurable data files that PSpice reads	11
Figure 2	Diode clipper circuit.	16
Figure 3	Connection points.	19
Figure 4	PSpice simulation output window.	22
Figure 5	Simulation output file.	24
Figure 6	DC sweep analysis settings.	27
Figure 7	Probe window.	28
Figure 8	Clipper circuit with voltage marker on net Out.	29
Figure 9	Voltage at In, Mid, and Out.	29
Figure 11	Trace legend with cursors activated.	30
Figure 12	Trace legend with V(Mid) symbol outlined.	30
Figure 13	Voltage difference at V(In) = 4 volts.	31
Figure 14	Diode clipper circuit with a voltage stimulus.	32
Figure 15	Stimulus Editor window.	34
Figure 16	Transient analysis simulation settings.	34
Figure 17	Sinusoidal input and clipped output waveforms.	35
Figure 18	Clipper circuit with AC stimulus.	37
Figure 19	AC sweep and noise analysis simulation settings.	38
Figure 20	dB magnitude curves for “gain” at Mid and Out.	40
Figure 21	Bode plot of clipper’s frequency response.	41
Figure 22	Clipper circuit with global parameter Rval.	42
Figure 23	Parametric simulation settings.	44
Figure 24	Small signal response as R1 is varied from 100 Ω to 10 k Ω	45
Figure 25	Small signal frequency response at 100 and 10 k Ω input resistance.	47
Figure 26	Performance analysis plots of bandwidth and gain vs. Rval.	50
Figure 27	Relationship of the Model Editor to Capture and PSpice	93
Figure 28	Process and data flow for the Model Editor.	96
Figure 29	Model Editor workspace with data for a bipolar transistor.	97
Figure 30	Design for a half-wave rectifier.	104
Figure 31	Model characteristics and parameter values for DbreakX.	105
Figure 32	Assorted device characteristic curves for a diode.	108
Figure 33	Forward Current device curve at two temperatures.	109

Figure 34	Rules for pin callout in subcircuit templates.	146
Figure 35	LOPASS filter example.	157
Figure 36	HIPASS filter part example.	158
Figure 37	BANDPASS filter part example.	159
Figure 38	BANDREJ filter part example.	159
Figure 39	FTABLE part example.	162
Figure 40	LAPLACE part example one.	165
Figure 41	Viewing gain and phase characteristics of a lossy integrator.	165
Figure 42	LAPLACE part example two.	165
Figure 43	ABM expression part example one.	169
Figure 44	ABM expression part example two.	169
Figure 45	ABM expression part example three.	170
Figure 46	ABM expression part example four.	170
Figure 47	Triode circuit.	171
Figure 48	Triode subcircuit producing a family of I-V curves.	173
Figure 49	EVALUE part example.	177
Figure 50	GVALUE part example.	177
Figure 51	EMULT part example.	178
Figure 52	GMULT part example.	179
Figure 53	EFREQ part example.	185
Figure 54	Voltage multiplier circuit (mixer).	186
Figure 55	PSpice simulation window	210
Figure 56	Example schematic EXAMPLE.OPJ.	217
Figure 57	Curve family example schematic.	221
Figure 58	Device curve family.	222
Figure 59	Operating point determination for each member of the curve family.	222
Figure 60	Circuit diagram for EXAMPLE.OPJ.	237
Figure 61	AC analysis setup for EXAMPLE.OPJ.	238
Figure 62	Device and total noise traces for EXAMPLE.DSN.	247
Figure 63	Transient analysis setup for EXAMPLE.OPJ.	263
Figure 64	Example schematic EXAMPLE.OPJ.	264
Figure 65	ECL-compatible Schmitt trigger.	266
Figure 66	Netlist for Schmitt trigger circuit.	267
Figure 67	Hysteresis curve example: Schmitt trigger.	268
Figure 68	Passive filter schematic.	274
Figure 69	Current of L1 when R1 is 1.5 ohms.	276
Figure 70	Rise time and overshoot vs. damping resistance.	277
Figure 71	RLC filter example circuit.	278
Figure 72	Plot of capacitance versus bias voltage.	280
Figure 73	Example schematic EXAMPLE.OPJ.	282
Figure 74	Example schematic EXAMPLE.DSN.	288
Figure 75	Monte Carlo analysis setup for EXAMPLE.DSN.	290

Figure 76	Summary of Monte Carlo runs for EXAMPLE.OPJ.	291
Figure 77	Parameter values for Monte Carlo pass three.	292
Figure 78	Pressure sensor circuit.	293
Figure 79	Model definition for RMonte1.	298
Figure 80	Pressure sensor circuit with RMonte1 and RTherm model definitions. .	299
Figure 81	Chebyshev filter.	302
Figure 82	1 dB bandwidth histogram.	305
Figure 83	Center frequency histogram.	306
Figure 84	Simple biased BJT amplifier.	309
Figure 85	Amplifier netlist and circuit file.	310
Figure 86	YatX Goal Function.	311
Figure 87	Correct worst-case results.	312
Figure 88	Incorrect worst-case results.	312
Figure 89	Schematic using VARY BOTH.	313
Figure 90	Circuit file using VARY BOTH.	313
Figure 91	Analog and digital areas of a plot.	321
Figure 92	Two Probe windows.	322
Figure 93	Trace legend symbols.	338
Figure 94	Section information message box.	339
Figure 95	Example schematic EXAMPLE.OPJ.	341
Figure 96	Waveform display for EXAMPLE.DAT.	342
Figure 97	Cursors positioned on a trough and peak of V(1)	352
Figure 98	Waveform display for a persistent hazard.	355
Figure A-1	Setpoints.	376

Tables

Table 1	DC analysis types	3
Table 2	AC analysis types	4
Table 3	Time-based analysis types	5
Table 4	Parametric and temperature analysis types	6
Table 5	Statistical analysis types	7
Table 2-1		25
Table 10	Association of cursors with mouse buttons.	30
Table 2-1		31
Table 2-1		36
Table 2-2		41
Table 2-3		48
Table 2-4		51
Table 5		58
Table 6		59
Table 7	Passive parts	64
Table 8	Breakout parts	65
Table 9	Operators in expressions	70
Table 10	Functions in arithmetic expressions	71
Table 11	System variables	73
Table 12		74
Table 13		75
Table 14		77
Table 15		78
Table 16		80
Table 17	Models supported in the Model Editor	95
Table 1	Sample diode data sheet values	106
Table 2	Part names for custom part generation.	133
Table 3		139
Table 4		141
Table 5		142
Table 6		145
Table 7	Control system parts	153

Table 1	ABM math function parts	167
Table 2	ABM expression parts	168
Table 1	PSpice -equivalent parts	174
Table 1	Basic controlled sources in ANALOG.OLB	192
Table 2	Classes of PSpice analyses	196
Table 3	Execution order for standard analyses	198
Table 4	PSpice output variable formats	201
Table 5	Element definitions for 2-terminal devices	202
Table 6	Element definitions for 3- or 4-terminal devices	203
Table 7	Element definitions for transmission line devices	204
Table 8	Element definitions for AC analysis specific elements	205
Table 9	DC sweep circuit design requirements	214
Table 10		218
Table 11		218
Table 12		219
Table 1	Curve family example setup	221
Table 2		233
Table 3		233
Table 4		234
Table 5		234
Table 6		236
Table 7		242
Table 8		244
Table 9		246
Table 10	Stimulus symbols for time-based input signals	252
Table 1	Parametric analysis circuit design requirements	272
Table 1	Collating functions used in statistical analyses	287
Table 1		294
Table 2		295
Table 3		300
Table 1	Default waveform viewing colors.	325
Table 2		326
Table 3		328
Table 4		330
Table 5		332
Table 6		333
Table 1		346
Table 2	Mouse actions for cursor control	351
Table 3	Key combinations for cursor control	351
Table 4		357
Table 5		358
Table 6	Output variable formats	358

Table 7	Examples of output variable formats	360
Table 8	Output variable AC suffixes	361
Table 9	Device names for two-terminal device types	361
Table 10	Terminal IDs by three & four-terminal device type	362
Table 11	Noise types by device type	363
Table 12	Analog arithmetic functions for trace expressions	364
Table 13	Output units for trace expressions	366
Table 14		369
Table 15		370

Before you begin

Welcome to OrCAD

OrCAD® offers a total solution for your core design tasks: schematic- and VHDL-based design entry; FPGA and CPLD design synthesis; digital, analog, and mixed-signal simulation; and printed circuit board layout. What's more, OrCAD's products are a suite of applications built around an engineer's design flow--not just a collection of independently developed point tools. PSpice is just one element in OrCAD's total solution design flow.

With OrCAD's products, you'll spend less time dealing with the details of tool integration, devising workarounds, and manually entering data to keep files in sync. Our products will help you build better products faster, and at lower cost.

OrCAD PSpice overview

OrCAD PSpice simulates analog-only circuits. After you prepare a design for simulation, OrCAD Capture generates a circuit file set. The circuit file set, containing the circuit netlist and analysis commands, is read by PSpice for simulation. PSpice formulates these into meaningful graphical plots, which you can mark for display directly from your schematic page using markers.

How to use this guide

This guide is designed so you can quickly find the information you need to use PSpice.

This guide assumes that you are familiar with Microsoft Windows (NT or 95), including how to use icons, menus, and dialog boxes. It also assumes you have a basic understanding about how Windows manages applications and files to perform routine tasks, such as starting applications, and opening and saving your work. If you are new to Windows, please review your *Microsoft Windows User's Guide*.

Typographical conventions

Before using PSpice, it is important to understand the terms and typographical conventions used in this documentation.

This guide generally follows the conventions used in the *Microsoft Windows User's Guide*. Procedures for performing an operation are generally numbered with the following typographical conventions..

Notation	Examples	Description
<code>Ctrl+R</code>	Press <code>Ctrl+R</code>	A specific key or key stroke on the keyboard.
monospace font	Type VAC . . .	Commands/text entered from the keyboard.

Related documentation

Documentation for OrCAD products is available in both printed and online forms. To access an online manual instantly, you can select it from the Help menu in its respective program (for example, access the Capture User’s Guide from the Help menu in Capture).

Note *The documentation you receive depends on the software configuration you have purchased.*

The following table provides a brief description of those manuals available in both printed and online forms.

This manual...	Provides information about how to use...
OrCAD Capture User’s Guide	OrCAD Capture, which is a schematic capture front-end program with a direct interface to other OrCAD programs and options.
OrCAD Layout User’s Guide	OrCAD Layout, which is a PCB layout editor that lets you specify printed circuit board sturcture, as well as the components, metal, and graphics required for fabrication.
OrCAD PSpice & Basics User’s Guide	PSpice with Probe, the Stimulus Editor, and the Model Editor, which are circuit analysis programs that let you create, simulate, and test analog and digital circuit designs. This manual provides examples on how to specify simulation parameters, analyze simulation results, edit input signals, and create models. (PSpice Basics is a limited version that does not include the Stimulus Editor.)
OrCAD PSpice User’s Guide	OrCAD PSpice with Probe is a circuit analysis program that lets you create, simulate, and test analog-only circuit designs.
OrCAD PSpice Optimizer User’s Guide	OrCAD PSpice Optimizer, which is an analog performance optimization program that lets you fine-tune your analog circuit designs.

The following table provides a brief description of those manuals available online *only*.

This online manual...	Provides this...
OrCAD PSpice Online Reference Manual	Reference material for PSpice. Also included: detailed descriptions of the simulation controls and analysis specifications, start-up option definitions, and a list of device types in the analog and digital model libraries. User interface commands are provided to instruct you on each of the screen commands.
OrCAD Application Notes Online Manual	A variety of articles that show you how a particular task can be accomplished using OrCAD's products, and examples that demonstrate a new or different approach to solving an engineering problem.
OrCAD PSpice Library List	A complete list of the analog and digital parts in the model and part libraries.

Online Help

Choosing Search for Help On from the Help menu displays an extensive online help system.

The online help includes:

- step-by-step instructions on how to set up PSpice simulations and analyze simulation results
- reference information about PSpice
- Technical Support information

If you are not familiar with Windows (NT or 95) Help system, choose How to Use Help from the Help menu.

If you have the demo CD-ROM

OrCAD demo CD-ROM

The OrCAD demo CD-ROM has the following limitations for PSpice:

- circuit simulation limited to circuits with up to 64 nodes, 10 transistors, two operational amplifiers or 65 digital primitive devices, and 10 transmission lines (ideal or non-ideal) with not more than 4 pairwise coupled lines
- device characterization using the Model Editor limited to diodes
- stimulus generation limited to sine waves (analog) and clocks (digital)
- sample library of approximately 39 analog and 134 digital parts
- displays only simulation data created using the demo version of the simulator
- PSpice Optimizer limited to one goal, one parameter and one constraint
- designs created in Capture can be saved if they have no more than 30 part instances

What's New

New PSpice interface with integrated waveform analysis functionality Release 9 of PSpice includes all of Probe's features and adds to them. Included in one screen are tabbed windows for viewing plots, text windows for viewing output files or other text files, and a simulation status and message window. Also included is a new, self-documenting analysis setup dialog for creating simulation profiles (see below). PSpice now provides an editable simulation queue which shows you how many files are currently in line to be simulated. You can edit or re-order the list as needed. And the plotting features have been improved by providing user-controlled grid settings, grid and trace properties (style and color) and metafile format copy and paste functions.

To find out more, see [Analyzing waveforms on page -319](#).

Simulation profiles PSpice Release 9 introduces the concept of simulation profiles. Each simulation profile refers to one schematic in a design and includes one analysis type (AC, DC, or Transient) with any options (sensitivity, temperature, parametric, Monte Carlo, etc.). You can define as many profiles as you need for your design and you can set up multiple analyses of the same type. Simulation profiles help you keep your analysis results separate, so you can delete one without losing the rest.

New OrCAD Capture front-end Release 9 integrates OrCAD Capture as the front-end schematic entry tool for PSpice. Capture provides a professional design entry environment with many advanced capabilities that now work hand-in-hand with PSpice. These include a project manager, a new property editor spreadsheet, right mouse button support, and many other time-saving features.

To find out more, see [Creating and editing models on page -85](#).

To find out more, refer to MOSFET devices in the *Analog Devices* chapter of the online *OrCAD PSpice A/D Reference Manual*.

New Model Editor interface The Model Editor (formerly known as Parts) has been improved and modernized for Release 9. It now provides a unified application for editing models either in text form or by modifying their specifications. The Model Editor now also supports Darlington modeling.

EKV version 2.6 MOSFET model The EKV model is a scalable and compact model built on fundamental physical properties of the device. Use this model to design low-voltage, low-current analog, and mixed analog-digital circuits that use sub-micron technologies.

Version 2.6 models the following:

- geometrical and process related aspects of the device (oxide thickness, junction depth, effective channel length and width, and so on)
- effects of doping profile and substrate effects
- weak, moderate, and strong inversion behavior
- mobility effects due to vertical and lateral fields and carrier velocity saturation
- short-channel effects such as channel-length modulation, source and drain charge sharing, and the reverse short channel effect
- thermal and flicker noise modeling
- short-distance geometry and bias-dependent device matching for Monte Carlo analysis

Enhanced model libraries The model libraries supplied with PSpice Release 9 have been enhanced to include the latest models from various vendors, as well as models for popular optocouplers, Darlingtons, and DAC and ADC devices.

Part one

Simulation primer

Part one provides basic information about circuit simulation including examples of common analyses.

- [Chapter 1, Things you need to know](#), provides an overview of the circuit simulation process including what PSpice does, descriptions of analysis types, and descriptions of important files.
- [Chapter 2, Simulation examples](#), presents examples of common analyses to introduce the methods and tools you'll need to enter, simulate, and analyze your design.

Things you need to know

1

Chapter overview

This chapter introduces the purpose and function of the OrCAD® PSpice circuit simulator.

- [What is PSpice? on page 1-2](#) describes PSpice capabilities.
- [Analyses you can run with PSpice on page 1-3](#) introduces the different kinds of basic and advanced analyses that PSpice supports.
- [Using PSpice with other OrCAD programs on page 1-9](#) presents the high-level simulation design flow.
- [Files needed for simulation on page 1-10](#) describes the files used to pass information between OrCAD programs. This section also introduces the things you can do to customize where and how PSpice finds simulation information.
- [Files that PSpice generates on page 1-14](#) describes the files that contain simulation results.

What is PSpice?

OrCAD PSpice is a simulation program that models the behavior of a circuit containing analog devices. Used with OrCAD Capture for design entry, you can think of PSpice as a software-based breadboard of your circuit that you can use to test and refine your design before ever touching a piece of hardware.

Run basic and advanced analyses PSpice can perform:

- DC, AC, and transient analyses, so you can test the response of your circuit to different inputs.
- Parametric, Monte Carlo, and sensitivity/worst-case analyses, so you can see how your circuit's behavior varies with changing component values.

The range of models built into PSpice include not only those for resistors, inductors, capacitors, and bipolar transistors, but also these:

- transmission line models, including delay, reflection, loss, dispersion, and crosstalk
- nonlinear magnetic core models, including saturation and hysteresis
- six MOSFET models, including BSIM3 version 3.1 and EKV version 2.6
- five GaAsFET models, including Parker-Skellern and TriQuint's TOM2 model
- IGBTs

Use parts from OrCAD's extensive set of libraries

The model libraries feature over 11,300 analog models of devices manufactured in North America, Japan, and Europe.

Vary device characteristics without creating new parts PSpice has numerous built-in models with parameters that you can tweak for a given device. These include independent temperature effects.

Model behavior PSpice supports analog behavioral modeling, so you can describe functional blocks of circuitry using mathematical expressions and functions.

Analyses you can run with PSpice

Basic analyses

DC sweep & other DC calculations

These DC analyses evaluate circuit performance in response to a direct current source. [Table 1](#) summarizes what PSpice calculates for each DC analysis type.

Table 1 *DC analysis types*

For this DC analysis...	PSpice computes this...
DC sweep	Steady-state voltages and currents when sweeping a source, a model parameter, or temperature over a range of values.
Bias point detail	Bias point data in addition to what is automatically computed in any simulation.
DC sensitivity	Sensitivity of a net or part voltage as a function of bias point.
Small-signal DC transfer	Small-signal DC gain, input resistance, and output resistance as a function of bias point.

See [Chapter 2, Simulation examples](#), for introductory examples showing how to run each type of analysis.

See [Part three, Setting Up and Running Analyses](#), for a more detailed discussion of each type of analysis and how to set it up.

AC sweep and noise

These AC analyses evaluate circuit performance in response to a small-signal alternating current source. [Table 2](#) summarizes what PSpice calculates for each AC analysis type.

Table 2 AC analysis types

For this AC analysis...	PSpice computes this...
AC sweep	Small-signal response of the circuit (linearized around the bias point) when sweeping one or more sources over a range of frequencies. Outputs include voltages and currents with magnitude and phase; you can use this information to obtain Bode plots.
Noise	For each frequency specified in the AC analysis: <ul style="list-style-type: none">• Propagated noise contributions at an output net from every noise generator in the circuit.• RMS sum of the noise contributions at the output.• Equivalent input noise.

Note *To run a noise analysis, you must also run an AC sweep analysis.*

Transient and Fourier

These time-based analyses evaluate circuit performance in response to time-varying sources. [Table 3](#) summarizes what PSpice calculates for each time-based analysis type.

Table 3 *Time-based analysis types*

For this time-based analysis...	PSpice computes this...
Transient	Voltages and currents tracked over time.
Fourier	DC and Fourier components of the transient analysis results.

Note *To run a Fourier analysis, you must also run a transient analysis.*

Advanced multi-run analyses

The multi-run analyses—parametric, temperature, Monte Carlo, and sensitivity/worst-case—result in a series of DC sweep, AC sweep, or transient analyses depending on which basic analyses you enabled.

Parametric and temperature

For parametric and temperature analyses, PSpice steps a circuit value in a sequence that you specify and runs a simulation for each value.

Table 4 shows the circuit values that you can step for each kind of analysis.

Table 4 *Parametric and temperature analysis types*

For this analysis...	You can step one of these...
Parametric	global parameter model parameter component value DC source operational temperature
Temperature	operational temperature

Monte Carlo and sensitivity/worst-case

Monte Carlo and sensitivity/worst-case analyses are statistical. PSpice changes device model parameter values with respect to device and lot tolerances that you specify, and runs a simulation for each value.

Table 5 summarizes how PSpice runs each statistical analysis type.

Table 5 *Statistical analysis types*

For this statistical analysis...	PSpice does this...
Monte Carlo	For each simulation, <i>randomly</i> varies all device model parameters for which you have defined a tolerance.
Sensitivity/ worst-case	Computes the probable worst-case response of the circuit in two steps: <ol style="list-style-type: none">1 Computes component sensitivity to changes in the device model parameters. This means PSpice <i>nonrandomly</i> varies device model parameters for which you have defined a tolerance, one at a time for each device and runs a simulation with each change.2 Sets <i>all</i> model parameters for <i>all</i> devices to their worst-case values (assumed to be at one of the tolerance limits) and runs a final simulation.

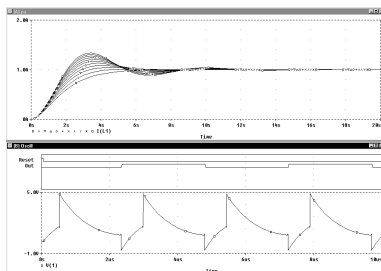
Analyzing waveforms with PSpice

What is waveform analysis?

After completing the simulation, PSpice plots the waveform results so you can visualize the circuit's behavior and determine the validity of your design.

Taken together, simulation and waveform analysis is an iterative process. After analyzing simulation results, you can refine your design and simulation settings and then perform a new simulation and waveform analysis.

Perform post-simulation analysis of the results This means you can plot additional information derived from the waveforms. What you can plot depends on the types of analyses you run. Bode plots, phase margin, derivatives for small-signal characteristics, waveform families, and histograms are only a few of the possibilities. You can also plot other waveform characteristics such as rise time versus temperature, or percent overshoot versus component value.



Using PSpice with other OrCAD programs

Using Capture to prepare for simulation

Capture is a design entry program you need to prepare your circuit for simulation. This means:

- placing and connecting part symbols,
- defining component values and other attributes,
- defining input waveforms,
- enabling one or more analyses, and
- marking the points in the circuit where you want to see results.

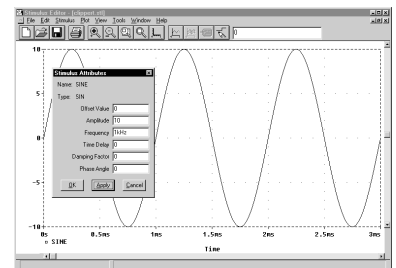
Capture is also the control point for running other programs used in the simulation design flow.

What is the Stimulus Editor?

The Stimulus Editor is a graphical input waveform editor that lets you define the shape of time-based signals used to test your circuit's response during simulation. Using the Stimulus Editor, you can define:

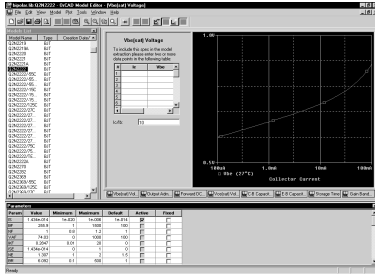
- analog stimuli with sine wave, pulse, piecewise linear, exponential pulse, single-frequency FM shapes

The Stimulus Editor lets you draw analog piecewise linear stimuli by clicking at the points along the timeline that correspond to the input values that you want at transitions.



What is the Model Editor?

The Model Editor is a model extractor that generates model definitions for PSpice to use during simulation. All the Model Editor needs is information about the device found in standard data sheets. As you enter the data sheet information, the Model Editor displays device characteristic curves so you can verify the model-based behavior of the device. When you are finished, the Model Editor automatically creates a part for the model so you can use the modeled part in your design immediately.



Files needed for simulation

To simulate your design, PSpice needs to know about:

- the parts in your circuit and how they are connected,
- what analyses to run,
- the simulation models that correspond to the parts in your circuit, and
- the stimulus definitions to test with.

This information is provided in various data files. Some of these are generated by Capture, others come from libraries (which can also be generated by other programs like the Stimulus Editor and the Model Editor), and still others are user-defined.

Files that Capture generates

When you begin the simulation process, Capture first generates files describing the parts and connections in your circuit. These files are the netlist file and the circuit file that PSpice reads before doing anything else.

Netlist file

The netlist file contains a list of device names, values, and how they are connected with other devices. The name that Capture generates for this file is DESIGN_NAME.NET.

Refer to the online *OrCAD PSpice Reference Manual* for the syntax of the statements in the netlist file and the circuit file.

Circuit file

The circuit file contains commands describing how to run the simulation. This file also refers to other files that contain netlist, model, stimulus, and any other user-defined information that apply to the simulation. The name that Capture generates for this file is DESIGN_NAME.CIR.

Other files that you can configure for simulation

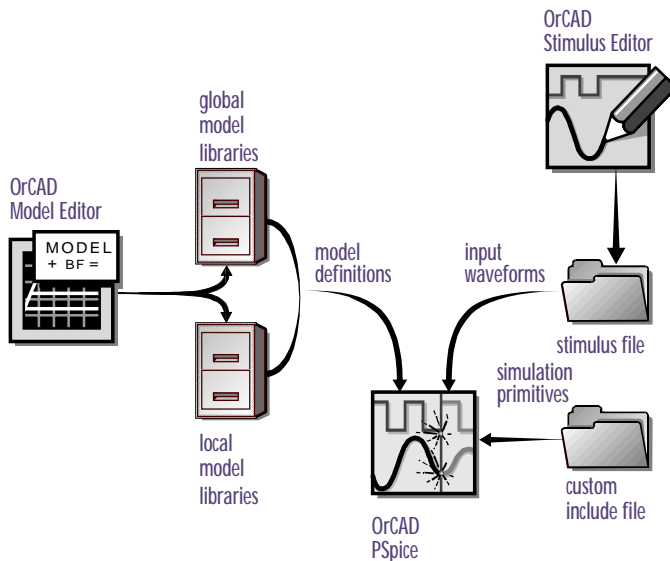


Figure 1 User-configurable data files that PSpice reads

Before starting simulation, PSpice needs to read other files that contain simulation information for your circuit. These are model files, and if required, stimulus files and include files.

The circuit file (.CIR) that Capture generates contains references to the other user-configurable files that PSpice needs to read.

You can create these files using OrCAD programs like the Stimulus Editor and the Model Editor. These programs automate file generation and provide graphical ways to verify the data. You can also use the Model Text view in the Model Editor (or another text editor like Notepad) to enter the data manually.

Model library

A model library is a file that contains the electrical definition of one or more parts. PSpice uses this information to determine how a part will respond to different electrical inputs.

These definitions take the form of either a:

- *model parameter set*, which defines the behavior of a part by fine-tuning the underlying model built into PSpice, or
- *subcircuit netlist*, which describes the structure and function of the part by interconnecting other parts and primitives.

The most commonly used models are available in the OrCAD model libraries shipped with your programs. The model library names have a .LIB extension.

If needed, however, you can create your own models and libraries, either:

- manually using the Model Text view in the Model Editor (or another text editor like Notepad), or
- automatically using the Model Editor.

A subcircuit, sometimes called a *macromodel*, is analogous to a procedure call in a software programming language.

See [What is the Model Editor?](#) on page 1-10 for a description.

Stimulus file

A stimulus file contains time-based definitions for analog input waveforms. You can create a stimulus file either:

- manually using the Model Text View of the Model Editor (or a standard text editor) to create the definition (a typical file extension is .STM), or
- automatically using the Stimulus Editor (which generates a .STL file extension).

Note *Not all stimulus definitions require a stimulus file. In some cases, like DC and AC sources, you must use a schematic symbol and set its attributes.*

See [What is the Stimulus Editor? on page 1-9](#) for a description.

Include file

An include file is a user-defined file that contains:

- PSpice commands, or
- supplemental text comments that you want to appear in the PSpice output file (see page [1-14](#)).

You can create an include file using any text editor, such as Notepad. Typically, include file names have a .INC extension.

Example: An include file that contains definitions, using the PSpice .FUNC command, for functions that you want to use in numeric expressions elsewhere in your design.

Configuring model library, stimulus, and include files

PSpice searches model libraries, stimulus files, and include files for any information it needs to complete the definition of a part or to run a simulation.

The files that PSpice searches depend on how you configure your model libraries and other files. Much of the configuration is set up for you automatically, however, you can do the following yourself:

- Add and delete files from the configuration.
- Change the scope of a file: that is, whether the file applies to one design only (local) or to any design (global).
- Change the search order.

More on libraries...

Configuration for model libraries is similar to that for other libraries that Capture uses, including part libraries. To find out more, refer to your Capture user's guide.

Files that PSpice generates

After reading the circuit file, netlist file, model libraries, and any other required inputs, PSpice starts the simulation. As simulation progresses, PSpice saves results to two files—the data file and the PSpice output file.

For a description of how to display simulation results, see [Part four, Viewing results](#).

For a description of the waveform analyzer program, see [What is waveform analysis? on page 1-8](#).

There are two ways to add waveforms to the display:

- From within PSpice, by specifying trace expressions.
- From within Capture, by cross-probing.

Example: Each instance of a VPRINT1 symbol placed in your schematic causes PSpice to generate a table of voltage values for the connecting net, and to write the table to the PSpice output file.

Waveform data file

The data file contains simulation results that that can be displayed graphically. PSpice reads this file automatically and displays waveforms reflecting circuit response at nets, pins, and parts that you marked in your schematic (cross-probing). You can set up your design so PSpice displays the results as the simulation progresses or after the simulation completes.

After PSpice has read the data file and displays the initial set of results, you can add more waveforms and to perform post-simulation analysis of the data.

PSpice output file

The PSpice output file is an ASCII text file that contains:

- the netlist representation of the circuit,
- the PSpice command syntax for simulation commands and options (like the enabled analyses),
- simulation results, and
- warning and error messages for problems encountered during read-in or simulation.

Its content is determined by:

- the types of analyses you run,
- the options you select for running PSpice, and
- the simulation control symbols (like VPRINT1 and VPLOT1) that you place and connect to nets in your design.

Simulation examples

2

Chapter overview

The examples in this chapter provide an introduction to the methods and tools for creating circuit designs, running simulations, and analyzing simulation results. All analyses are performed on the same example circuit to clearly illustrate analysis setup, simulation, and result-analysis procedures for each analysis type.

This chapter includes the following sections:

- [Example circuit creation on page 2-16](#)
- [Performing a bias point analysis on page 2-22](#)
- [DC sweep analysis on page 2-26](#)
- [Transient analysis on page 2-32](#)
- [AC sweep analysis on page 2-37](#)
- [Parametric analysis on page 2-42](#)
- [Performance analysis on page 2-49](#)

Example circuit creation

This section describes how to use OrCAD Capture to create the simple diode clipper circuit shown in Figure 2.

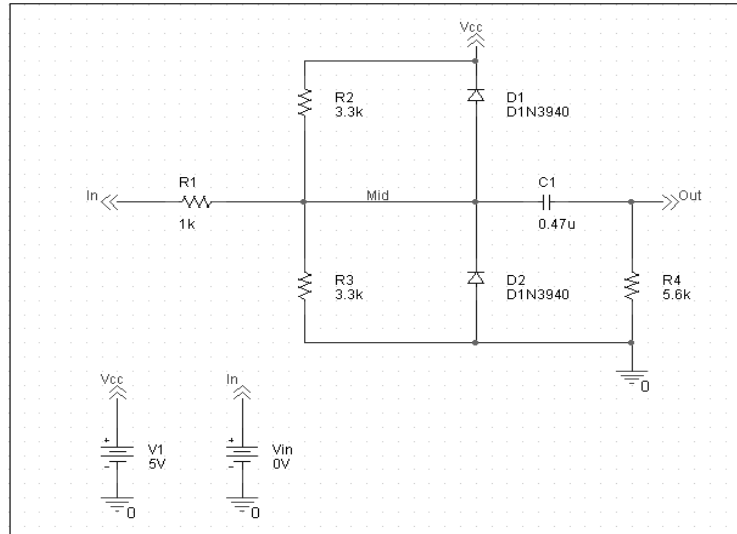


Figure 2 Diode clipper circuit.

To create a new PSpice project

- 1 From the Windows Start menu, choose the OrCAD Release 9 program folder and then the Capture shortcut to start Capture.
- 2 In the Project Manager, from the File menu, point to New and choose Project.
- 3 Select Analog Circuit Wizard.
- 4 In the Name text box, enter the name of the project (CLIPPER).
- 5 Click OK, then click Finish.

No special libraries need to be configured at this time. A new page will be displayed in Capture and the new project will be configured in the Project Manager.

To place the voltage sources

- 1 In Capture, switch to the schematic page editor.

- 2 From the Place menu, choose Part to display the Place Part dialog box.
- 3 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select SOURCE.OLB (from the PSpice library) and click Open.
- 4 In the Part text box, type VDC.
- 5 Click OK.
- 6 Move the pointer to the correct position on the schematic page (see Figure 2) and click to place the first part.
- 7 Move the cursor and click again to place the second part.
- 8 Right-click and choose End Mode to stop placing parts.



Note There are two sets of library files supplied with Capture and PSpice. The standard schematic part libraries are found in the directory Capture\Library. The part libraries that are designed for simulation with PSpice are found in the sub-directory Capture\Library\PSpice. In order to have access to specific parts, you must first configure the library in Capture using the Add Library function.

To place the diodes

- 1 From the Place menu, choose Part to display the Place Part dialog box.
- 2 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select DIODE.OLB (from the PSpice library) and click Open.
- 3 In the Part text box, type D1N39 to display a list of diodes.
- 4 Select D1N3940 and click OK.
- 5 Press **[R]** to rotate the diode to the correct orientation.
- 6 Click to place the first diode (D1), then click to place the second diode (D2).
- 7 Right-click and choose End Mode to stop placing parts.



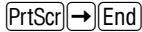
When placing parts:

- Leave space to connect the parts with wires.
- You will change part names and values that do not match those shown in Figure 2 later in this section.

To move the text associated with the diodes (or any other object)

- 1 Click the text to select it, then drag the text to a new location.

To place the other parts



- 1 From the Place menu, choose Part to display the Place Part dialog box.
- 2 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select ANALOG.OLB (from the PSpice library) and click Open.
- 3 Follow similar steps as described for the diodes to place the parts listed below, according to Figure 2. The part names you need to type in the Part name text box of the Place Part dialog box are shown in parentheses:
 - resistors (R)
 - capacitor (C)



- 4 To place the off-page connector parts (OFFPAGELEFT-R), click the Place Off-Page Connector button on the tool palette.
- 5 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select CAPSYM.OLB (from the Capture library) and click Open.



- 6 Place the off-page connector parts according to Figure 2.
- 7 To place the ground parts (0), click the GND button on the tool palette.
- 8 Add the library for the parts you need to place:
 - a Click the Add Library button.
 - b Select SOURCE.OLB (from the PSpice library) and click Open.
- 9 Place the ground parts according to Figure 2.

To connect the parts

- 1 From the Place menu, choose Wire to begin wiring parts.
The pointer changes to a crosshair.
- 2 Click the connection point (the very end) of the pin on the off-page connector at the input of the circuit.
- 3 Click the nearest connection point of the input resistor R1.
- 4 Connect the other end of R1 to the output capacitor.
- 5 Connect the diodes to each other and to the wire between them:
 - a Click the connection point of the cathode for the lower diode.
 - b Move the cursor straight up and click the wire between the diodes. The wire ends, and the junction of the wire segments becomes visible.
 - c Click again on the junction to continue wiring.
 - d Click the end of the upper diode's anode pin.
- 6 Continue connecting parts until the circuit is wired as shown in Figure 2 on page 2-16.

To assign names (labels) to the nets

- 1 From the Place menu, choose Net Alias to display the Place Net Alias dialog box.
- 2 In the Name text box, type Mid.
- 3 Click OK.
- 4 Place the net alias on any segment of the wire that connects R1, R2, R3, the diodes, and the capacitor. The lower left corner of the net alias must touch the wire.
- 5 Right-click and choose End Mode to quit the Net Alias function.



To stop wiring, right-click and choose End Wire. The pointer changes to the default arrow.

Clicking on any valid connection point ends a wire. A valid connection point is shown as a *box* (see Figure 3).

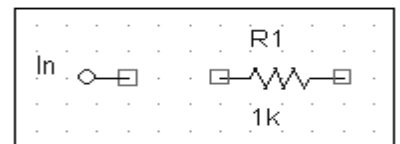


Figure 3 Connection points.

If you make a mistake when placing or connecting components:

- 1 From the Edit menu, choose Undo, or click .

To assign names (labels) to the off-page connectors

Label the off-page connectors as shown in Figure 2 on page 2-16.

- 1 Double-click the name of an off-page connector to display the Display Properties dialog box.
- 2 In the Name text box, type the new name.
- 3 Click OK.
- 4 Select and relocate the new name as desired.

To assign names to the parts

A more efficient way to change the names, values and other properties of several parts in your design is to use the Property Editor, as follows:

- 1 Select all of the parts to be modified by pressing **Ctrl** and clicking each part.
- 2 From the Edit menu, choose Properties.

The Parts Spreadsheet appears.

Change the entries in as many of the cells as needed, and then click Apply to update all of the changes at once.

- 1 Double-click the second VDC part to display the Parts spreadsheet.
- 2 Click in the first cell under the Reference column.
- 3 Type in the new name Vin.
- 4 Click Apply to update the changes to the part, then close the spreadsheet.
- 5 Continue naming the remaining parts until your schematic looks like Figure 2 on page 2-16.

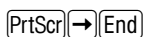
To change the values of the parts

- 1 Double-click the voltage label (0V) on V1 to display the Display Properties dialog box.
- 2 In the Value text box, type 5v.
- 3 Click OK.
- 4 Continue changing the Part Value properties of the parts until all the parts are defined as in Figure 2 on page 2-16.

Your schematic page should now have the same parts, wiring, labels, and properties as Figure 2 on page 2-16.

To save your design

- 1 From the File menu, choose Save.



Finding out more about setting up your design

About setting up a design for simulation

For a checklist of all of the things you need to do to set up your design for simulation, and how to avoid common problems, see [Chapter 3, Preparing a design for simulation](#).

Running PSpice

When you perform a simulation, PSpice generates an output file (*.OUT).

You can set up a simulation profile to run one analysis at a time. To run multiple analyses (for example, both DC sweep and transient analyses), set up a batch simulation. For more information, see [Chapter 7, Setting up analyses and starting simulation](#).

While PSpice is running, the progress of the simulation appears and is updated in the PSpice simulation output window (see Figure 4).

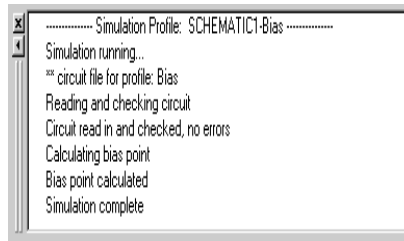


Figure 4 PSpice simulation output window.

Performing a bias point analysis

To set up a bias point analysis in Capture

- 1 In Capture, switch to CLIPPER.OPJ in the schematic page editor.
- 2 From the PSpice menu, choose New Simulation Profile to display the New Simulation dialog box.
- 3 In the Name text box, type Bias.
- 4 From the Inherit From list, select None, then click Create.
The Simulation Settings dialog box appears.
- 5 From the Analysis type list, select Bias Point.
- 6 Click OK to close the Simulation Settings dialog box.

The root schematic listed is the schematic page associated with the simulation profile you are creating.

To simulate the circuit from within Capture

- 1 From the PSpice menu, choose Run.



PSpice simulates the circuit and calculates the bias point information.

Note *Because waveform data is not calculated during a bias point analysis, you will not see any plots displayed in the Probe window for this simulation. To find out how to view the results of this simulation, see Using the simulation output file below.*

Using the simulation output file

The simulation output file acts as an audit trail of the simulation. This file optionally echoes the contents of the circuit file as well as the results of the bias point calculation. If there are any syntax errors in the netlist declarations or simulation commands, or anomalies while performing the calculation, PSpice writes error or warning messages to the output file.

To view the simulation output file

- 1 From PSpice's View menu, choose Output File.

Figure 5 shows the results of the bias point calculation as written in the simulation output file.

```

**** 10/05/98 12:06:18 ***** PSpice 9.0 (Aug 1998) ***** ID# 3 *****

** circuit file for profile: Bias

****    SMALL SIGNAL BIAS SOLUTION      TEMPERATURE =  27.000 DEG C

*****

NODE   VOLTAGE   NODE   VOLTAGE   NODE   VOLTAGE   NODE   VOLTAGE

(  IN)  0.0000  (  OUT)  0.0000  (  VCC)  5.0000  (M00744)  .9434

VOLTAGE SOURCE CURRENTS
NAME      CURRENT
V_V1      -1.229E-03
V_Vin      9.434E-04

TOTAL POWER DISSIPATION  6.15E-03 WATTS

```

Figure 5 *Simulation output file.*

- 2 When finished, close the window.

PSpice measures the current through a two terminal device into the first terminal and out of the second terminal. For voltage sources, current is measured from the positive terminal to the negative terminal; this is opposite to the positive current flow convention and results in a negative value in the output file.

Finding out more about bias point calculations

Table 2-1

To find out more about this...	See this...
bias point calculations	Bias point on page 8-223

DC sweep analysis

You can visually verify the DC response of the clipper by performing a DC sweep of the input voltage source and displaying the waveform results in the Probe window in PSpice. This example sets up DC sweep analysis parameters to sweep V_{in} from -10 to 15 volts in 1 volt increments.

Setting up and running a DC sweep analysis

To set up and run a DC sweep analysis

- 1 In Capture, from the PSpice menu, choose New Simulation Profile.
The New Simulation dialog box appears.
- 2 In the Name text box, type DC Sweep.
- 3 From the Inherit From list, select Schematic1-Bias, then click Create.
The Simulation Settings dialog box appears.
- 4 Click the Analysis tab.
- 5 From the Analysis type list, select DC Sweep and enter the values shown in Figure 6.

Note The default settings for DC Sweep simulation are Voltage Source as the swept variable type and Linear as the sweep type. To use a different swept variable type or sweep type, choose different options under Sweep variable and Sweep type.

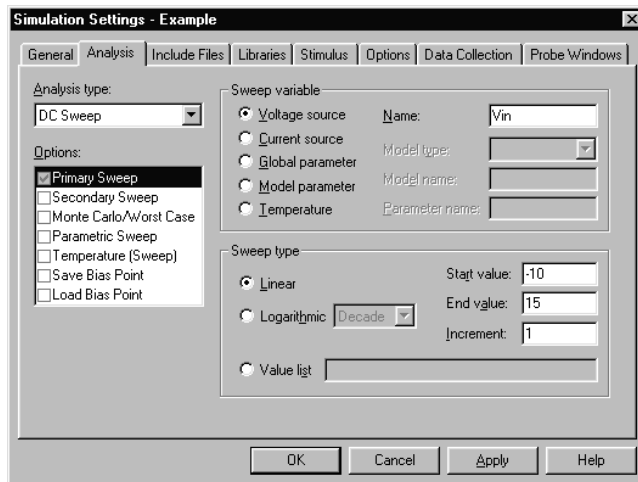


Figure 6 *DC sweep analysis settings.*

- 6 Click OK to close the Simulation Settings dialog box.
- 7 From the File menu, choose Save.
- 8 From the PSpice menu, choose Run to run the analysis.



Displaying DC analysis results

Probe windows can appear during or after the simulation is finished.

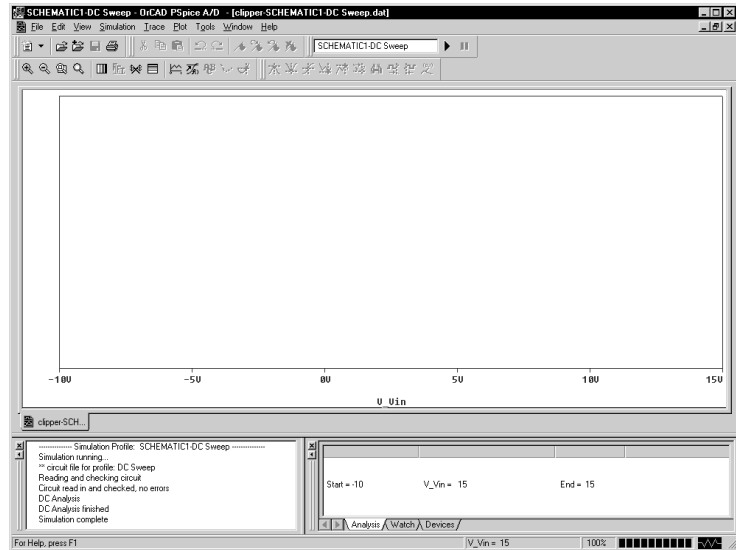


Figure 7 *Probe window.*

To plot voltages at nets In and Mid

press **Insert**



- 1 From PSpice's Trace menu, choose Add Trace.
- 2 In the Add Traces dialog box, select V(In) and V(Mid).
- 3 Click OK.

To display a trace using a marker

press **Ctrl + M**

- 1 From Capture's PSpice menu, point to Markers and choose Voltage Level.
- 2 Click to place a marker on net Out, as shown in Figure 8.

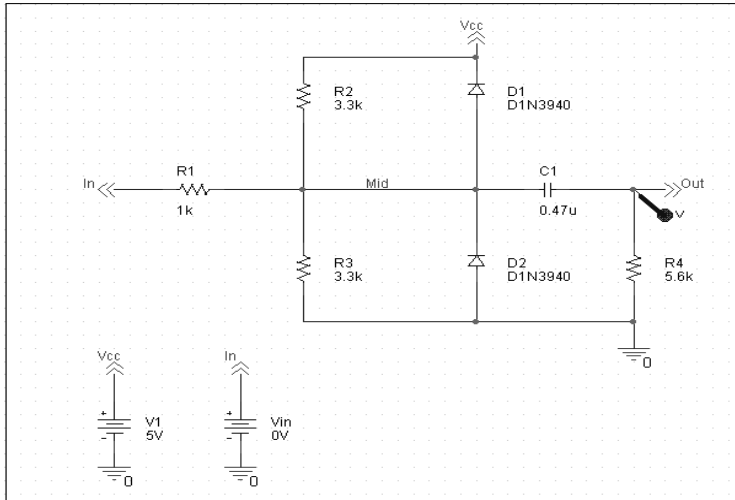


Figure 8 *Clipper circuit with voltage marker on net Out.*

- 3 Right-click and choose End Mode to stop placing markers.
- 4 From the File menu, choose Save.
- 5 Switch to PSpice. The V(Out) waveform trace appears, as shown in Figure 9.

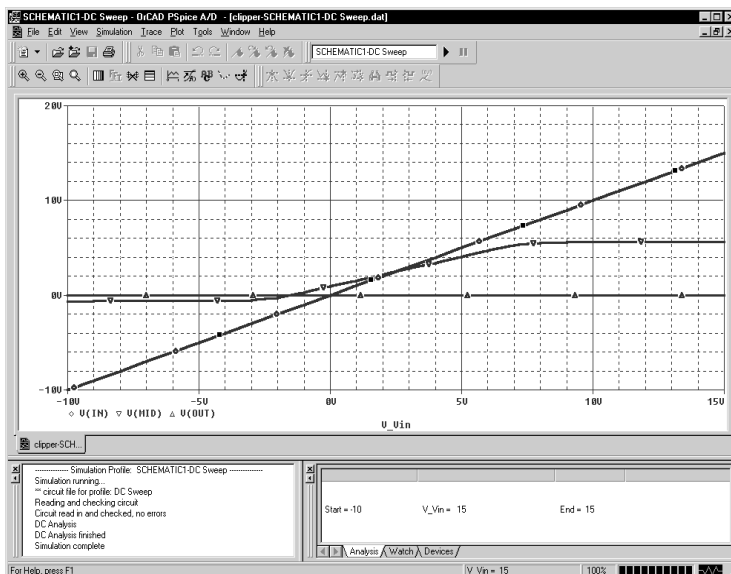


Figure 9 *Voltage at In, Mid, and Out.*

This example uses the cursors feature to view the numeric values for two traces and the difference between them by placing a cursor on each trace.

Table 10 Association of cursors with mouse buttons.

cursor 1	left mouse button
cursor 2	right mouse button

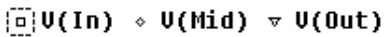


Figure 11 Trace legend with cursors activated.



Your ability to get as close to 4.0 as possible depends on screen resolution and window size.



Figure 12 Trace legend with V(Mid) symbol outlined.

To place cursors on V(In) and V(Mid)

- 1 From PSpice's Trace menu, point to Cursor and choose Display.

Two cursors appear for the first trace defined in the legend below the x-axis—V(In) in this example. The Probe Cursor window also appears.
- 2 To display the cursor crosshairs:
 - a Position the mouse anywhere inside the Probe window.
 - b Click to display the crosshairs for the first cursor.
 - c Right-click to display the crosshairs for the second cursor.
In the trace legend, the part for V(In) is outlined in the crosshair pattern for each cursor, resulting in a dashed line as shown in Figure 11.
- 3 Place the first cursor on the V(In) waveform:
 - a Click the portion of the V(In) trace in the proximity of 4 volts on the x-axis. The cursor crosshair appears, and the current X and Y values for the first cursor appear in the cursor window.
 - b To fine-tune the cursor location to 4 volts on the x-axis, drag the crosshairs until the x-axis value of the A1 cursor in the cursor window is approximately 4.0. You can also press  and  for tighter control.
- 4 Place the second cursor on the V(Mid) waveform:
 - a Right-click the trace legend part (diamond) for V(Mid) to associate the second cursor with the Mid waveform. The crosshair pattern for the second cursor outlines the V(Mid) trace part as shown in Figure 12.
 - b Right-click the portion on the V(Mid) trace that is in the proximity of 4 volts on the x-axis. The X and Y values for the second cursor appear in the cursor window along with the difference (dif) between the two cursors' X and Y values.

- c To fine-tune the location of the second cursor to 4 volts on the x-axis, drag the crosshairs until the x-axis value of the A2 cursor in the cursor window is approximately 4.0. You can also press **Shift**+**→** and **Shift**+**←** for tighter control.

Figure 13 shows the Probe window with both cursors placed.

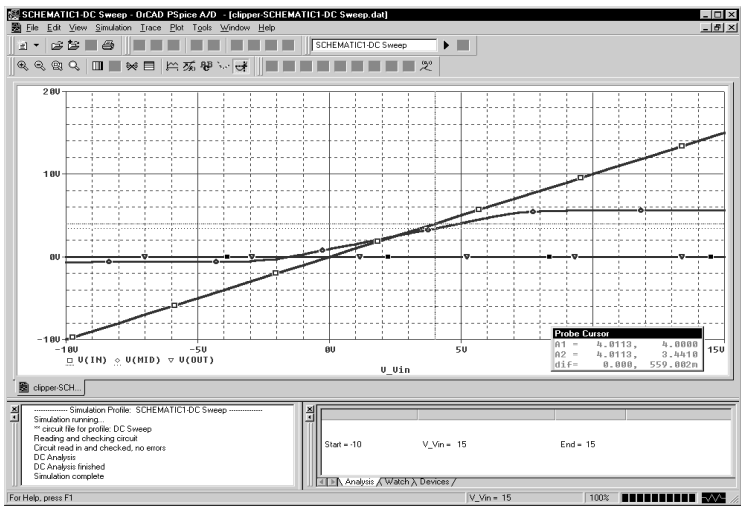


Figure 13 Voltage difference at $V(In) = 4$ volts.

To delete all of the traces

- 1 From the Trace menu, choose Delete All Traces.
At this point, the design has been saved. If needed, you can quit Capture and PSpice and complete the remaining analysis exercises later using the saved design.

There are also ways to display the difference between two voltages as a trace:

- In PSpice, add the trace expression $V(In)-V(Mid)$.
- In Capture, from the PSpice menu, point to Markers and choose Voltage Differential. Place the two markers on different pins or wires.

You can also delete an individual trace by selecting its name in the trace legend and then pressing **Delete**.

Example: To delete the $V(In)$ trace, click the text, $V(In)$, located under the plot's x-axis, and then press **Delete**.

Finding out more about DC sweep analysis

Table 2-1

To find out more about this...	See this...
DC sweep analysis	DC Sweep on page 8-214

Transient analysis

This example shows how to run a transient analysis on the clipper circuit. This requires adding a time-domain voltage stimulus as shown in Figure 14.

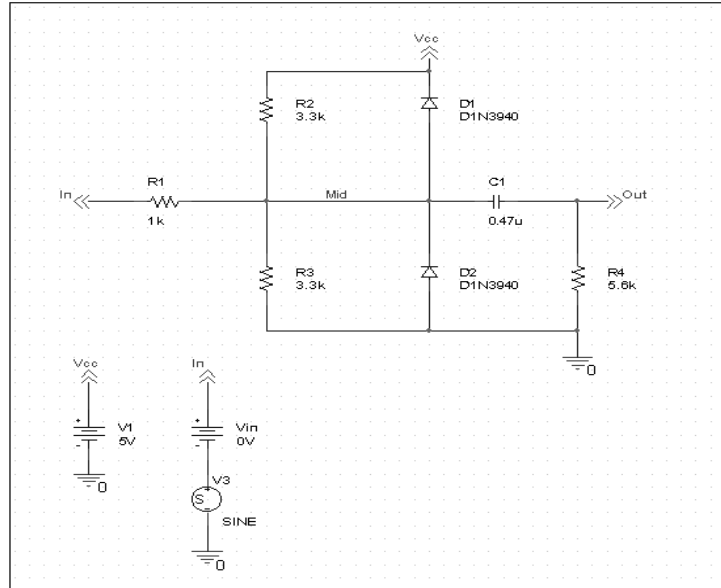


Figure 14 *Diode clipper circuit with a voltage stimulus.*

To add a time-domain voltage stimulus

- 1 From Capture's PSpice menu, point to Markers and choose Delete All.
- 2 Select the ground part beneath the VIN source.
- 3 From the Edit menu, choose Cut.
- 4 Scroll down (or from the View menu, point to Zoom, then choose Out).
- 5 Place a VSTIM part (from the PSpice library SOURCESTM.OLB) as shown in Figure 14.
- 6 From the Edit menu, choose Paste.
- 7 Place the ground part under the VSTIM part as shown in Figure 14.
- 8 From the View menu, point to Zoom, then choose All.
- 9 From the File menu, choose Save to save the design.



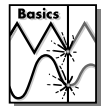
or press **Ctrl** + **V**

To set up the stimulus

- 1 Select the VSTIM part (V3).
- 2 From the Edit menu, choose PSpice Stimulus.
The New Stimulus dialog box appears.
- 3 In the New Stimulus dialog box, type `SINE`.
- 4 Click SIN (sinusoidal), then click OK.
- 5 In the SIN Attributes dialog box, set the first three properties as follows:
 Offset Voltage = 0
 Amplitude = 10
 Frequency = 1kHz
- 6 Click Apply to view the waveform.

The Stimulus Editor window should look like Figure 15.

Note The Stimulus Editor is not included in PSpice Basics.



If you do not have the Stimulus Editor

- 1 Place a VSIN part instead of VSTIM and double-click it.
- 2 In the Edit Part dialog box, click User Properties.
- 3 Set values for the VOFF, VAMPL, and FREQ properties as defined in step 5. When finished, click OK.

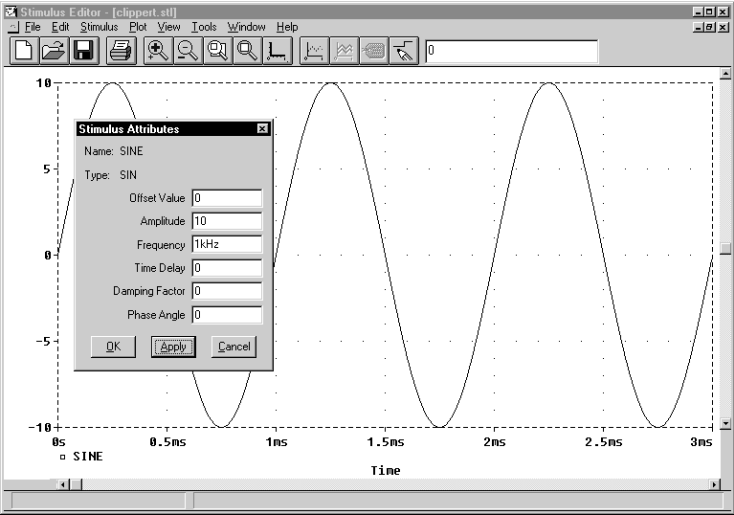


Figure 15 *Stimulus Editor window.*

7 Click OK.

press **Shift** + **F12**



8 From the File menu, choose Save to save the stimulus information. Click Yes to update the schematic.

9 From the File menu, choose Exit to exit the Stimulus Editor.

To set up and run the transient analysis

1 From Capture’s PSpice menu, choose New Simulation Profile.

The New Simulation dialog box appears.

2 In the Name text box, type Transient.

3 From the Inherit From list, select Schematic1-DC Sweep, then click Create.

The Simulation Settings dialog box appears.

4 Click the Analysis tab.

5 From the Analysis list, select Time Domain (Transient) and enter the settings shown in Figure 16.

TSTOP = 2ms

Start saving data after = 20ns

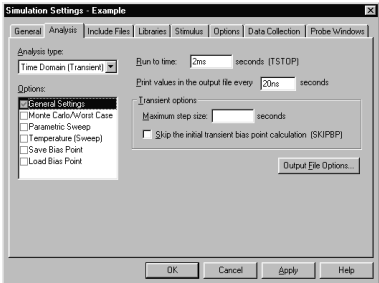


Figure 16 *Transient analysis simulation settings.*

- 6 Click OK to close the Simulation Settings dialog box.
- 7 From the PSpice menu, choose Run to perform the analysis.

PSpice uses its own internal time steps for computation. The internal time step is adjusted according to the requirements of the transient analysis as it proceeds. PSpice saves data to the waveform data file for each internal time step.



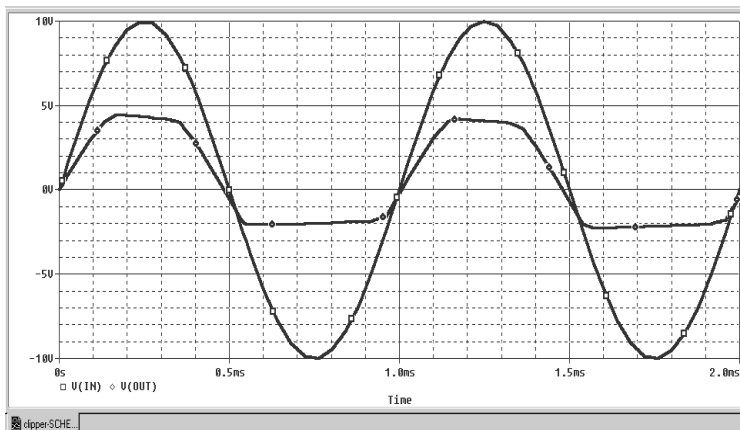
Note The internal time step is different from the Print Step value. Print Step controls how often optional text format data is written to the simulation output file (*.OUT).

To display the input sine wave and clipped wave at V(Out)

- 1 From PSpice's Trace menu, choose Add Trace.
- 2 In the trace list, select V(In) and V(Out) by clicking them.
- 3 Click OK to display the traces.
- 4 From the Tools menu, choose Options to display the Probe Options dialog box.
- 5 In the Use Symbols frame, click Always if it is not already enabled.
- 6 Click OK.



or press **Insert**



These waveforms illustrate the clipping of the input signal.

Figure 17 Sinusoidal input and clipped output waveforms.

Finding out more about transient analysis

Table 2-1

To find out more about this...	See this...
transient analysis for analog designs*	Chapter 10, Transient analysis

* Includes how to set up time-based stimuli using the Stimulus Editor.

AC sweep analysis

The AC sweep analysis in PSpice is a linear (or small signal) frequency domain analysis that can be used to observe the frequency response of any circuit at its bias point.

Setting up and running an AC sweep analysis

In this example, you will set up the clipper circuit for AC analysis by adding an AC voltage source for a stimulus signal (see Figure 18) and by setting up AC sweep parameters.

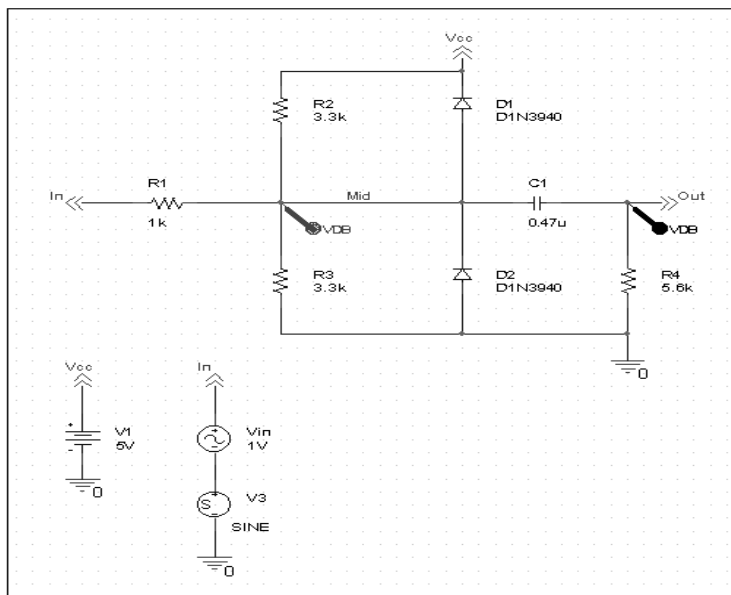


Figure 18 *Clipper circuit with AC stimulus.*

To change Vin to include the AC stimulus signal

- 1 In Capture, open CLIPPER.OPJ.
- 2 Select the DC voltage source, Vin, and press Delete to remove the part from the schematic page.

- 3 From the Place menu, choose Part.
- 4 In the Part text box, type `VAC` (from the PSpice library SOURCE.OLB) and click OK.
- 5 Place the AC voltage source on the schematic page, as shown in Figure 17.
- 6 Double-click the VAC part (0V) to display the Parts spreadsheet.
- 7 Change the Reference cell to `Vin` and change the ACMAG cell to `1V`.
- 8 Click Apply to update the changes and then close the spreadsheet.

To set up and run the AC sweep simulation

Note PSpice simulation is not case-sensitive, so both *M* and *m* can be used as “milli,” and *MEG*, *Meg*, and *meg* can all be used for “mega.” However, waveform analysis treats *M* and *m* as mega and milli, respectively.

- 1 From Capture’s PSpice menu, choose New Simulation Profile.
- 2 In the Name text box, enter AC Sweep, then click create.

The Simulation Settings dialog box appears.

- 3 Click the Analysis tab.
- 4 From the Analysis type list, select AC Sweep/Noise and enter the settings shown in Figure 19.

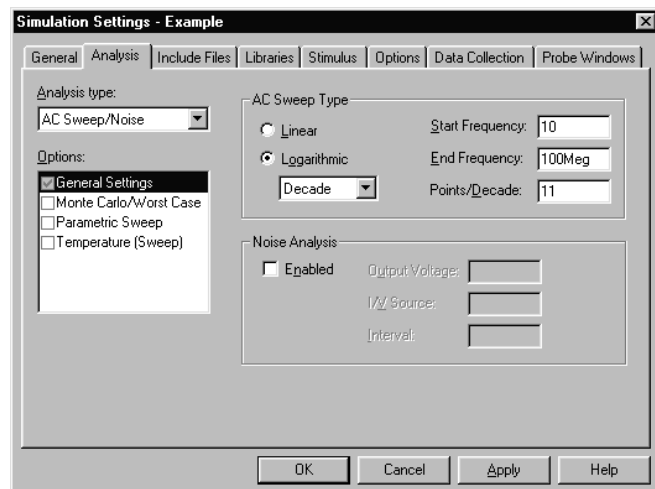


Figure 19 AC sweep and noise analysis simulation settings.

- 5 Click OK to close the Simulation Settings dialog box.
- 6 From the PSpice menu, choose Run to start the simulation.



PSpice performs the AC analysis.

To add markers for waveform analysis

- 1 From Capture's PSpice menu, point to Markers, point to Advanced, then choose db Magnitude of Voltage.
- 2 Place one Vdb marker on the Out net, then place another on the Mid net.
- 3 From the File menu, choose Save to save the design.

Note You must first define a simulation profile for the AC Sweep/Noise analysis in order to use advanced markers.

AC sweep analysis results

PSpice displays the dB magnitude ($20\log_{10}$) of the voltage at the marked nets, Out and Mid, in a Probe window as shown in Figure 20 below. VDB(Mid) has a lowpass response due to the diode capacitances to ground. The output capacitance and load resistor act as a highpass filter, so the overall response, illustrated by VDB(out), is a bandpass response. Because AC is a linear analysis and the input voltage was set to 1V, the output voltage is the same as the gain (or attenuation) of the circuit.

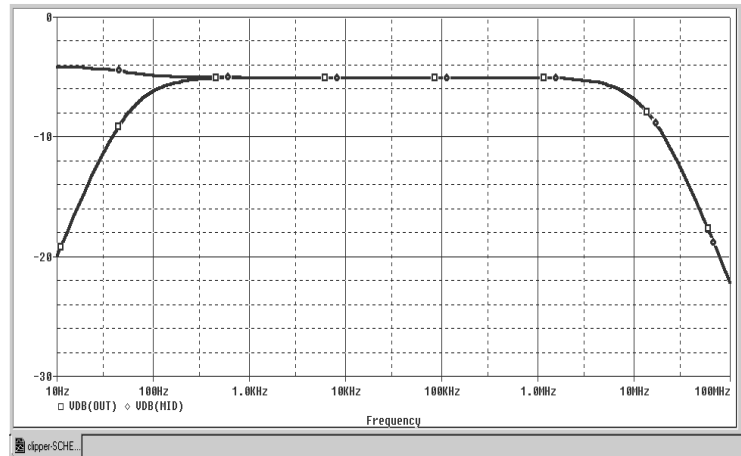


Figure 20 dB magnitude curves for “gain” at Mid and Out.

Note Depending upon where the Vphase marker was placed, the trace name may be different, such as VP(Cout:2), VP(R4:1), or VP(R4:2).

For more information on Probe windows and trace expressions, see [Chapter 13, Analyzing waveforms](#).

press **Ctrl** + **X**



press **Ctrl** + **V**



To display a Bode plot of the output voltage, including phase

- 1 From Capture's PSpice menu, point to Markers, point to Advanced and choose Phase of Voltage.
- 2 Place a Vphase marker on the output next to the Vdb marker.
- 3 Delete the Vdb marker on Mid.
- 4 Switch to PSpice.

In the Probe window, the gain and phase plots both appear on the same graph with the same scale.

- 5 Click the trace name VP(Out) to select the trace.
- 6 From the Edit menu, choose Cut.
- 7 From the Plot menu, choose Add Y Axis.
- 8 From the Edit menu, choose Paste.

The Bode plot appears, as shown in Figure 21.

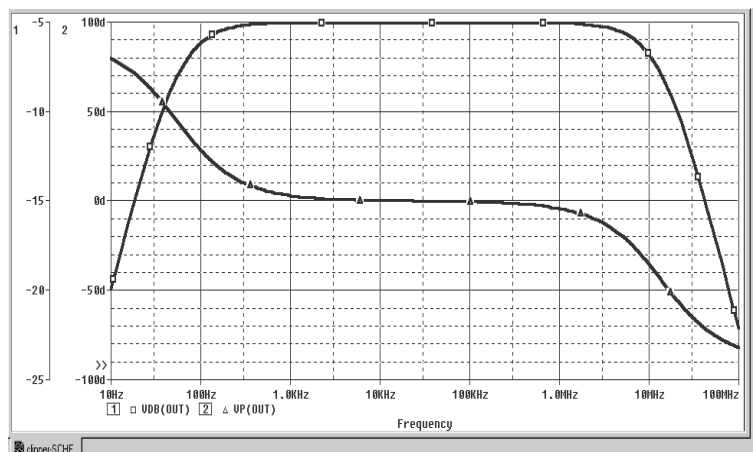
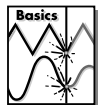


Figure 21 Bode plot of clipper’s frequency response.

Finding out more about AC sweep and noise analysis

Table 2-2

To find out more about this...	See this...
AC sweep analysis	AC sweep analysis on page 9-232
noise analysis based on an AC sweep analysis	Noise analysis on page 9-241



Note *Parametric Analysis is not supported in PSpice Basics.*

Parametric analysis

This example shows the effect of varying input resistance on the bandwidth and gain of the clipper circuit by:

- Changing the value of R1 to the expression {Rval}.
- Placing a PARAM part to declare the parameter Rval.
- Setting up and running a parametric analysis to step the value of R1 using Rval.

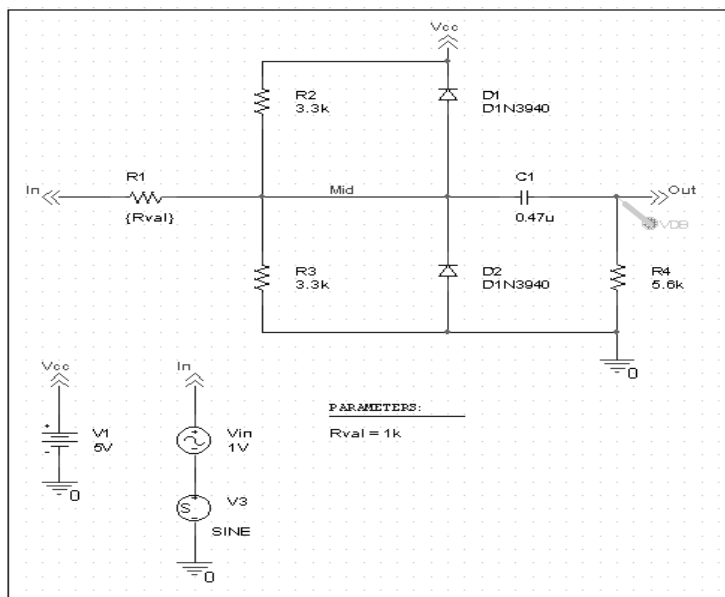


Figure 22 *Clipper circuit with global parameter Rval.*

This example produces multiple analysis runs, each with a different value of R1. After the analysis is complete, you can analyze curve families for the analysis runs using PSpice.

Setting up and running the parametric analysis

To change the value of R1 to the expression {Rval}

- 1 In Capture, open CLIPPER.OPI.
- 2 Double-click the value (1k) of part R1 to display the Display Properties dialog box.
- 3 In the Value text box, replace 1k with {Rval}.
- 4 Click OK.

PSpice interprets text in curly braces as an expression that evaluates to a numerical value. This example uses the simplest form of an expression—a constant. The value of R1 will take on the value of the Rval parameter, whatever it may be.

To add a PARAM part to declare the parameter Rval

- 1 From Capture's Place menu, choose Part.
- 2 In the Part text box, type `PARAM` (from the PSpice library SPECIAL.OLB) , then click OK.
- 3 Place one PARAM part in any open area on the schematic page.
- 4 Double-click the PARAM part to display the Parts spreadsheet, then click New.
- 5 In the Property Name text box, enter `Rval` (no curly braces), then click OK.

This creates a new property for the PARAM part, as shown by the new column labeled Rval in the spreadsheet.

- 6 Click in the cell below the Rval column and enter `1k` as the initial value of the parametric sweep.
- 7 While this cell is still selected, click Display.
- 8 In the Display Format frame, select Name and Value, then click OK.
- 9 Click Apply to update all the changes to the PARAM part.
- 10 Close the Parts spreadsheet.
- 11 Select the VP marker and press Delete to remove the marker from the schematic page.
- 12 From the File menu, choose Save to save the design.

Note For more information about using the Parts spreadsheet, see the OrCAD Capture User's Guide.

This example is only interested in the magnitude of the response.

To set up and run a parametric analysis to step the value of R1 using Rval

- 1 From Capture's PSpice menu, choose New Simulation Profile.

The New Simulation dialog box appears.

- 2 In the Name text box, type `Parametric`.
- 3 From the Inherit From list, select AC Sweep, then click Create.

The Simulation Settings dialog box appears.

- 4 Click the Analysis tab.
- 5 Under Options, select Parametric Sweep and enter the settings as shown below.

The root schematic listed is the schematic page associated with the simulation profile you are creating.

This profile specifies that the parameter `Rval` is to be stepped from 100 to 10k logarithmically with a resolution of 10 points per decade.

The analysis is run for each value of `Rval`. Because the value of `R1` is defined as `{Rval}`, the analysis is run for each value of `R1` as it logarithmically increases from 100 Ω to 10 k Ω in 20 steps, resulting in a total of 21 runs.

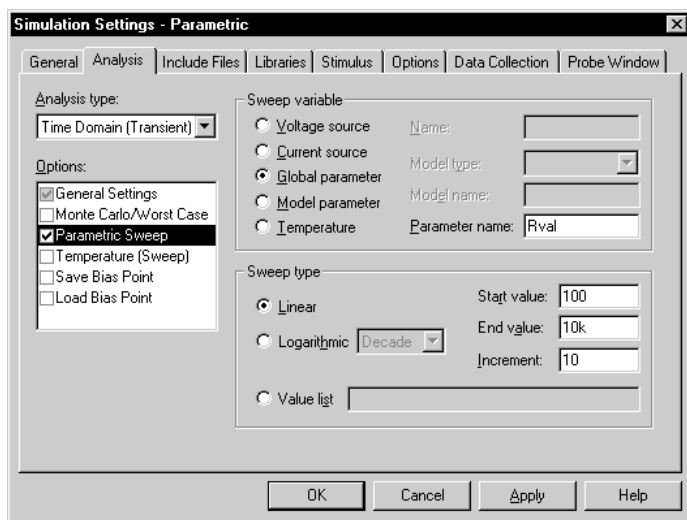


Figure 23 Parametric simulation settings.

- 6 Click OK.
- 7 From the PSpice menu, choose Run to start the analysis.



Analyzing waveform families

Continuing from the example above, there are 21 analysis runs, each with a different value of R1. After PSpice completes the simulation, the Available Sections dialog box appears, listing all 21 runs and the Rval parameter value for each. You can select one or more runs to display.

To display all 21 traces

- 1 In the Available Sections dialog box, click OK.
All 21 traces (the entire family of curves) for VDB(Out) appear in the Probe window as shown in Figure 24.

To select individual runs, click each one separately.

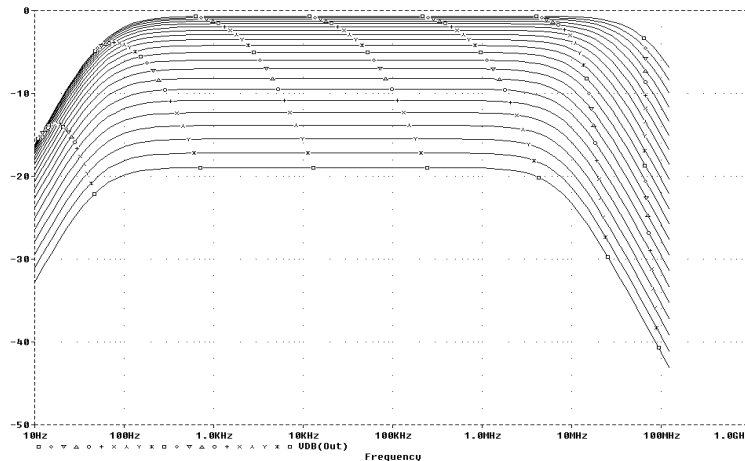


Figure 24 *Small signal response as R1 is varied from 100Ω to 10 kΩ*

To see more information about the section that produced a specific trace, double-click the corresponding symbol in the legend below the x-axis.

- 2 Click the trace name to select it, then press **Delete** to remove the traces shown.

You can also remove the traces by removing the VDB marker from your schematic page in Capture.

press **Insert**

You can avoid some of the typing for the Trace Expression text box by selecting V(OUT) twice in the trace list and inserting text where appropriate in the resulting Trace Expression.

press **Insert**

The search command tells PSpice to search for the point on the trace where the x-axis value is 100.

To compare the last run to the first run

- 1 From the Trace menu, choose Add Trace to display the Add Traces dialog box.
- 2 In the Trace Expression text box, type the following:
`Vdb(Out)@1 Vdb(Out)@21`
- 3 Click OK.

Note *The difference in gain is apparent. You can also plot the difference of the waveforms for runs 21 and 1, then use the search commands to find certain characteristics of the difference.*

- 4 Plot the new trace by specifying a waveform expression:
 - a From the Trace menu, choose Add Trace.
 - b In the Trace Expression text box, type the following waveform expression:
`Vdb(Out)@1-Vdb(OUT)@21`
 - c Click OK.
- 5 Use the search commands to find the value of the difference trace at its maximum and at a specific frequency:
 - a From the Tools menu, point to Cursor and choose Display.
 - b Right-click then left-click the trace part (triangle) for Vdb(Out)@1 - Vdb(Out)@21. Make sure that you left-click last to make cursor 1 the active cursor.
 - c From the Trace menu, point to Cursor and choose Max.
 - d From the Trace menu, point to Cursor and choose Search Commands.
 - e In the Search Command text box, type the following:
`search forward x value (100)`
 - f Select 2 as the Cursor to Move option.

g Click OK.

Figure 25 shows the Probe window with cursors placed.

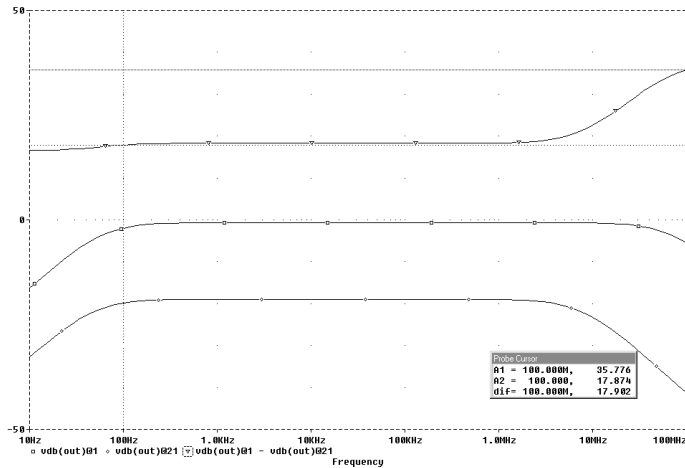


Figure 25 *Small signal frequency response at 100 and 10 k Ω input resistance.*

Note that the Y value for cursor 2 in the cursor box is about 17.87. This indicates that when R1 is set to 10 k Ω , the small signal attenuation of the circuit at 100Hz is 17.87dB greater than when R1 is 100 Ω .

- 6 From the Trace menu, point to Cursor and choose Display to turn off the display of the cursors.
- 7 Delete the trace.



Finding out more about parametric analysis

Table 2-3

To find out more about this...	See this...
parametric analysis	<u>Parametric analysis on page 11-272</u>
using global parameters	<u>Using global parameters and expressions for values on page 3-67</u>

Performance analysis

Performance analysis is an advanced feature in PSpice that you can use to compare the characteristics of a family of waveforms. Performance analysis uses the principle of search commands introduced earlier in this chapter to define functions that detect points on each curve in the family.

After you define these functions, you can apply them to a family of waveforms and produce traces that are a function of the variable that changed within the family.

This example shows how to use performance analysis to view the dependence of circuit characteristics on a swept parameter. In this case, the small signal bandwidth and gain of the clipper circuit are plotted against the swept input resistance value.

To plot bandwidth vs. Rval using the performance analysis wizard

- 1 In Capture, open CLIPPER.OPJ.
- 2 From PSpice's Trace menu, choose Performance Analysis.
The Performance Analysis dialog box appears with information about the currently loaded data and performance analysis in general.
- 3 Click the Wizard button.
- 4 Click the Next> button.
- 5 In the Choose a Goal Function list, click Bandwidth, then click the Next> button.
- 6 Click in the Name of Trace to search text box and type `V(Out)`.
- 7 Click in the db level down for bandwidth calc text box and type 3.
- 8 Click the Next> button.

At each step, the wizard provides information and guidelines.

Click , then double-click `V(Out)`.

The wizard displays the gain trace for the first run ($R=100$) and shows how the bandwidth is measured. This is done to test the goal function.

Double-click the x-axis.

- 9 Click the Next> button or the Finish button.
A plot of the 3dB bandwidth vs. Rval appears.
- 10 Change the x-axis to log scale:
 - a From the Plot menu, choose Axis Settings.
 - b Click the X Axis tab.
 - c Under Scale, choose Log.
 - d Click OK.

To plot gain vs. Rval manually



or press **Insert**

The Trace list includes goal functions only in performance analysis mode when the x-axis variable is the swept parameter.

- 1 From the Plot menu, choose Add Y Axis.
- 2 From the Trace menu, choose Add to display the Add Traces dialog box.
- 3 In the Functions or Macros frame, select the Goal Functions list, and then click the Max(1) goal function.
- 4 In the Simulation Output Variables list, click V(out).
- 5 In the Trace Expression text box, edit the text to be `Max(Vdb(out))`, then click OK.

PSpice displays gain on the second y-axis vs. Rval.

Figure 26 shows the final performance analysis plot of 3dB bandwidth and gain in dB vs. the swept input resistance value.

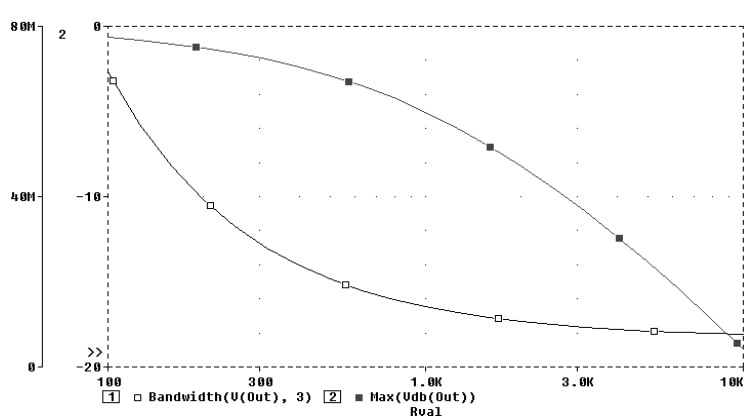


Figure 26 Performance analysis plots of bandwidth and gain vs. Rval.

Finding out more about performance analysis

Table 2-4

To find out more about this...	See this...
how to use performance analysis	RLC filter example on page 11-274 Example: Monte Carlo analysis of a pressure sensor on page 12-293
how to use search commands and create goal functions	PSpice A/D online Help

Part two

Design entry

Part two provides information about how to enter circuit designs in OrCAD® Capture that you want to simulate.

- [Chapter 3, Preparing a design for simulation](#), outlines the things you need to do to successfully simulate your schematic including troubleshooting tips for the most frequently asked questions.
- [Chapter 4, Creating and editing models](#), describes how to use the tools to create and edit model definitions, and how to configure the models for use.
- [Chapter 5, Creating parts for models](#), explains how to create symbols for existing or new model definitions so you can use the models when simulating from your schematic.
- [Chapter 6, Analog behavioral modeling](#), describes how to model analog behavior mathematically or using table lookups.

Preparing a design for simulation

3

Chapter overview

This chapter provides introductory information to help you enter circuit designs that simulate properly. If you want an overview, use the checklist on page [3-56](#) to guide you to specific topics.

Topics include:

- [Checklist for simulation setup on page 3-56](#)
- [Using parts that you can simulate on page 3-60](#)
- [Using global parameters and expressions for values on page 3-67](#)
- [Defining power supplies on page 3-74](#)
- [Defining stimuli on page 3-75](#)
- [Things to watch for on page 3-79](#)

Refer to your *OrCAD Capture User's Guide* for general schematic entry information.

Checklist for simulation setup

This section describes what you need to do to set up your circuit for simulation.

- 1 Find the topic that is of interest in the first column of any of these tables.
- 2 Go to the referenced section. For those sections that provide overviews, you will find references to more detailed discussions.

Typical simulation setup steps

For more information on this step...	See this...	To find out this...
✓ Set component values and other properties.	Using parts that you can simulate on page 3-60 Using global parameters and expressions for values on page 3-67	An overview of vendor, passive, breakout, and behavioral parts. How to define values using variable parameters, functional calls, and mathematical expressions.
✓ Define power supplies.	Defining power supplies on page 3-74	An overview of DC power for analog circuits..
✓ Define input waveforms.	Defining stimuli on page 3-75	An overview of DC, AC, and time-based stimulus parts.
✓ Set up one or more analyses.	Chapter 7, Setting up analyses and starting simulation Chapter 8 through Chapter 12 (see the table of contents)	Procedures, general to all analysis types, to set up and start the simulation. Detailed information about DC, AC, transient, parametric, temperature, Monte Carlo, and sensitivity/worst-case..

For more information on this step...	See this...	See this...
✓ Place markers.	Using schematic page markers to add traces on page 13-331	How to display results in PSpice by picking design nets.
✓	Limiting waveform data file size on page 13-334	How to limit the data file size.

Advanced design entry and simulation setup steps

For more information on this step...	See this...	To find out how to...
✓ Create new models.	Chapter 4, Creating and editing models	Define models using the Model Editor or Create Subcircuit command.
	Chapter 6, Analog behavioral modeling	Define the behavior of a block of analog circuitry as a mathematical function or lookup table.
✓ Create new parts.	Chapter 5, Creating parts for models	Create parts either automatically for models using the part wizard or the Parts utility, or by manually defining AKO parts; define simulation-specific properties.
	<i>The OrCAD Capture User's Guide</i>	Create and edit part graphics, pins, and properties in general.

When netlisting fails or the simulation does not start

If you have problems starting the simulation, there may be problems with the design or with system resources. If there are problems with the design, PSpice displays errors and warnings in the Simulation Output window. You can use the Simulation Output window to get more information quickly about the specific problem.

To get online information about an error or warning shown in the Simulation Output window

- 1 Select the error or warning message.
- 2 Press **[F1]**.

The following tables list the most commonly encountered problems and where to find out more about what to do.

Things to check in your design

Table 5

Make sure that...	To find out more, see this...
✓ The model libraries, stimulus files, and include files are configured.	Configuring model libraries on page 4-120
✓ The parts you are using have models.	Unmodeled parts on page 3-79 and Defining part properties needed for simulation on page 5-139
✓ You are not using unmodeled pins.	Unmodeled pins on page 3-82
✓ You have defined the grounds.	Missing ground on page 3-83
✓ Every analog net has a DC path to ground.	Missing DC path to ground on page 3-84
✓ The part template is correct.	Defining part properties needed for simulation on page 5-139
✓ Hierarchical parts, if used, are properly defined.	The <i>OrCAD Capture User's Guide</i>
✓ Ports that connect to the same net have the same name.	The <i>OrCAD Capture User's Guide</i>

Things to check in your system configuration

Table 6

Make sure that...	To find out more, see this...
✓ Path to the PSpice programs is correct.	
✓ Directory containing your design has write permission.	Your operating system manual
✓ Your system has sufficient free memory and disk space.	Your operating system manual

Using parts that you can simulate

The OrCAD part libraries also include special parts that you can use for simulation only. These include:

- **stimulus parts** to generate input signals to the circuit (see [Defining stimuli on page 3-75](#))
- **ground parts** required by all analog circuits, which need reference to ground
- **simulation control parts** to do things like set bias values (see [Appendix A, Setting initial state](#))
- **output control parts** to do things like generate tables and line-printer plots to the PSpice output file (see [Chapter 14, Other output options](#))

The OrCAD part libraries supply numerous parts designed for simulation. These include:

- vendor-supplied parts
- passive parts
- breakout parts
- behavioral parts

At minimum, a part that you can simulate has these properties:

- A simulation model to describe the part's electrical behavior; the model can be:
 - explicitly defined in a model library,
 - built into PSpice, or
 - built into the part (for some kinds of analog behavioral parts).
- A part with modeled pins to form electrical connections in your design.
- A translation from design part to netlist statement so that PSpice can read it in.

Note *Not all parts in the libraries are set up for simulation. For example, connectors are parts destined for board layout only and do not have these simulation properties.*

Vendor-supplied parts

The OrCAD libraries provide an extensive selection of manufacturers' analog parts. Typically, the library name reflects the kind of parts contained in the library and the vendor that provided the models.

Example: MOTOR_RF.OLB and MOTOR_RF.LIB contain parts and models, respectively, for Motorola-made RF bipolar transistors.

For a listing of vendor-supplied parts contained in the OrCAD libraries, refer to the online *Library List*.

To find out more about each model library, read the comments in the .LIB file header.

Part naming conventions

The part names in the OrCAD libraries usually reflect the manufacturers' part names. If multiple vendors supply the same part, each part name includes a suffix that indicates the vendor that supplied the model.

Example: The OrCAD libraries include several models for the OP-27 opamp as shown by these entries in the online *Library List*.

Device Type	Generic Name	Mfg. Name	Symbol Name	Library	Tech Type	Model	Package	New for 8.0
Operational Amplifier	OP-249	Analog Devices Inc.	OP-249G/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-260	Analog Devices Inc.	OP-260/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27A/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27B/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27C/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27E/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27F/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Analog Devices Inc.	OP-27G/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27	Linear Technology Corp.	OP-27/LT	LIN_TECH.SLB		*	*	
Operational Amplifier	OP-27		OP-27	OPAMP.SLB		*	*	
Operational Amplifier	OP-275	Analog Devices Inc.	OP-275/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-275	Analog Devices Inc.	OP-275G/AD	ANLG_DEV.SLB		*	*	
Operational Amplifier	OP-27A	Linear Technology Corp.	OP-27A/LT	LIN_TECH.SLB		*	*	
Operational Amplifier	OP-27C	Linear Technology Corp.	OP-27C/LT	LIN_TECH.SLB		*	*	

Notice the following:

- There is a generic OP-27 part provided by OrCAD, the OP-27/AD from Analog Devices, Inc., and the OP-27/LT from Linear Technology Corporation.
- The Model column for all of these parts contains an asterisk. This indicates that this part is modeled and that you can simulate it.

Finding the part that you want

If you are having trouble finding a part, you can search the libraries for parts with similar names by using either:

- the parts browser in Capture and restricting the parts list to those names that match a specified wildcard text string, or
- the online *Library List* and searching for the generic part name using capabilities of the Adobe Acrobat Reader.

To find parts using the parts browser

- 1 In Capture, from the Place menu, choose Part.
- 2 In the Part Name text box, type a text string with wildcards that approximates the part name that you want to find. Use this syntax:

`<wildcard><part_name_fragment><wildcard>`

where `<wildcard>` is one of the following:

- * to match zero or more characters
- ? to match exactly one character

The parts browser displays only the matching part names.

Note *This method finds any part contained in the current part libraries configuration, including parts for user-defined models.*

If you want to find out more about a part supplied in the OrCAD libraries, such as manufacturer or whether you can simulate it, then search the online *Library List* (see page [3-63](#)).

To find parts using the online OrCAD Library List

- 1 In Windows Explorer, double-click LIBLIST.PDF, located in the directory where PSpice is installed. Acrobat Reader starts and displays the OrCAD Library List.
- 2 From the Tools menu, choose Find.
- 3 In the Find What text box, type the generic part name.
- 4 Enter any other search criteria, and then click Find.
The Acrobat Reader displays the first page where it finds a match. Each page maps the generic part name to the parts (and corresponding vendor and part library name) in the OrCAD libraries.
- 5 If you want to repeat the search, from the Tools menu, choose Find Again.

Note *If you are unsure of the device type, you can scan all of the device type lists using the Acrobat search capability. The first time you do this, you need to set up the across-list index. To find out more, refer to the online Adobe Acrobat manuals.*

Note *This method finds only parts that OrCAD supplies that have models.*

If you want to include user-defined parts in the search, use the parts browser in Capture (see page [3-62](#)).



or press **Ctrl** + **F**

Instead of the generic part name, you can enter other kinds of search information, such as device type or manufacturer.

press **Ctrl** + **G**

Passive parts

The OrCAD libraries supply several basic parts based on the passive device models built into PSpice. These are summarized in the following table.

Table 7 *Passive parts*

To find out more about how to use these parts and define their properties, look up the corresponding PSpice device letter in the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*, and then see the *Capture Parts* sections.

These parts are available...	For this device type...	Which is this PSpice device letter...
C C_VAR	capacitor	C
L	inductor	L
R R_VAR	resistor	R
XFRM_LINEAR K_LINEAR	transformer	K and L
T	ideal transmission line	T
TLOSSY	Lossy transmission line	T
TnCOUPLED* TnCOUPLEDX** KCOUPLEn**	coupled transmission line	T and K

* For these device types, the OrCAD libraries supply several parts. Refer to the online *OrCAD PSpice A/D Reference Manual* for the available parts.

Breakout parts

The OrCAD libraries supply passive and semiconductor parts with default model definitions that define a basic set of model parameters. This way, you can easily:

- assign device and lot tolerances to model parameters for Monte Carlo and sensitivity/worst-case analyses,
- define temperature coefficients, and
- define device-specific operating temperatures.

These are called breakout parts and are summarized in the following table.

Table 8 Breakout parts

Use this breakout part...	For this device type...	Which is this PSpice device letter...
BBREAK	GaAsFET	B
CBREAK	capacitor	C
DBREAK _x *	diode	D
JBREAK _x *	JFET	J
KBREAK	inductor coupling	K
LBREAK	inductor	L
MBREAK _x *	MOSFET	M
QBREAK _x *	bipolar transistor	Q
RBREAK	resistor	R
SBREAK	voltage-controlled switch	S
TBREAK	transmission line	T
WBREAK	current-controlled switch	W
XFRM_NONLINEAR	transformer	K and L
ZBREAKN	IGBT	Z

* For this device type, the OrCAD libraries supply several breakout parts. Refer to the online *OrCAD PSpice Reference Manual* for the available parts.

To find out more about models, see [What are models? on page 4-87](#).

To find out more about Monte Carlo and sensitivity/worst-case analyses, see [Chapter 12, Monte Carlo and sensitivity/worst-case analyses](#).

To find out more about setting temperature parameters, see the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual* and find the device type that you are interested in.

To find out more about how to use these parts and define their properties, look up the corresponding PSpice device letter in the *Analog Devices* chapter of the online *OrCAD PSpice A/D Reference Manual*, and then look in the *Capture Parts* section.

Behavioral parts

Behavioral parts allow you to define how a block of circuitry should work without having to define each discrete component.

For more information, see [Chapter 6, Analog behavioral modeling](#).

Analog behavioral parts These parts use analog behavioral modeling (ABM) to define each part's behavior as a mathematical expression or lookup table. The OrCAD libraries provide ABM parts that operate as math functions, limiters, Chebyshev filters, integrators, differentiators, and others that you can customize for specific expressions and lookup tables. You can also create your own ABM parts.

Using global parameters and expressions for values

In addition to literal values, you can use global parameters and expressions to represent numeric values in your circuit design.

Global parameters

A global parameter is like a programming variable that represents a numeric value by name.

Once you have defined a parameter (declared its name and given it a value), you can use it to represent circuit values anywhere in the design; this applies to any hierarchical level.

Some ways that you can use parameters are as follows:

- Apply the same value to multiple part instances.
- Set up an analysis that sweeps a variable through a range of values (for example, DC sweep or parametric analysis).

When multiple parts are set to the same value, global parameters provide a convenient way to change all of their values for “what-if” analyses.

Example: If two independent sources have a value defined by the parameter VSUPPLY, then you can change both sources to 10 volts by assigning the value *once* to VSUPPLY.

Declaring and using a global parameter

To use a global parameter in your design, you need to:

- define the parameter using a PARAM part, and
- use the parameter in place of a literal value somewhere in your design.

Note For more information about using the Parts spreadsheet, see the *OrCAD Capture User's Guide*.

Example: To declare the global parameter VSUPPLY that will set the value of an independent voltage source to 14 volts, place the PARAM part, and then create a new property named VSUPPLY with a value of 14v.

To declare a global parameter

- 1 Place a PARAM part in your design.
- 2 Double-click the PARAM part to display the Parts spreadsheet, then click New.
- 3 Declare up to three global parameters by doing the following for each global parameter:
 - a Click New.
 - b In the Property Name text box, enter NAME n , then click OK.
This creates a new property for the PARAM part, NAME n in the spreadsheet.
 - c Click in the cell below the NAME n column and enter a default value for the parameter.
 - d While this cell is still selected, click Display.
 - e In the Display Format frame, select Name and Value, then click OK.
- Note** The system variables in [Table 11 on page 3-73](#) have reserved parameter names. Do not use these parameter names when defining your own parameters.
- 4 Click Apply to update all the changes to the PARAM part.
- 5 Close the Parts spreadsheet.

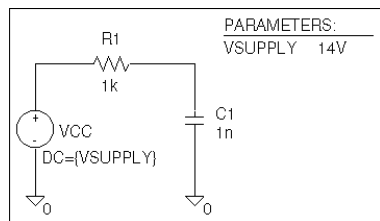
To use the global parameter in your circuit

- 1 Find the numeric value that you want to replace: a component value, model parameter value, or other property value.
- 2 Replace the value with the name of the global parameter using the following syntax:

`{ global_parameter_name }`

The curly braces tell PSpice to *evaluate* the parameter and use its value.

Example: To set the independent voltage source, VCC, to the value of the VSUPPLY parameter, set its DC property to {VSUPPLY}.



Expressions

An expression is a mathematical relationship that you can use to define a numeric or boolean (TRUE/FALSE) value.

PSpice evaluates the expression to a single value every time:

- it reads in a new circuit, and
- a parameter value used within an expression changes during an analysis.

Example: A parameter that changes with each step of a DC sweep or parametric analysis.

Specifying expressions

To use an expression in your circuit

- 1 Find the numeric or boolean value you want to replace: a component value, model parameter value, other property value, or logic in an IF function test (see page [3-72](#) for a description of the IF function).
- 2 Replace the value with an expression using the following syntax:

`{ expression }`

where *expression* can contain any of the following:

- standard operators listed in [Table 9](#)
- built-in functions listed in [Table 10](#)
- user-defined functions
- system variables listed in [Table 11](#)
- user-defined global parameters
- literal operands

The curly braces tell PSpice to *evaluate* the expression and use its value.

Example: Suppose you have declared a parameter named FACTOR (with a value of 1.2) and want to scale a -10 V independent voltage source, VEE, by the value of FACTOR. To do this, set the DC property of VEE to:

`{-10*FACTOR}`

PSpice evaluates this expression to:

`(-10 * 1.2) or -12 volts`

For more information on user-defined functions, see the .FUNC command in the *Commands* chapter in the online *OrCAD PSpice A/D Reference Manual*.

For more information on user-defined parameters, see [Using global parameters and expressions for values on page 3-67](#).

Table 9 *Operators in expressions*

This operator class...	Includes this operator...	Which means...
arithmetic	+	addition or string concatenation
	-	subtraction
	*	multiplication
	/	division
	**	exponentiation
logical*	~	unary NOT
		boolean OR
	^	boolean XOR
	&	boolean AND
relational*	==	equality test
	!=	non-equality test
	>	greater than test
	>=	greater than or equal to test
	<	less than test
	<=	less than or equal to test

* Logical and relational operators are used within the IF() function.

Table 10 *Functions in arithmetic expressions*

This function...	Means this...	
ABS(x)	$ x $	
SQRT(x)	$x^{1/2}$	
EXP(x)	e^x	
LOG(x)	$\ln(x)$	which is log base e
LOG10(x)	$\log(x)$	which is log base 10
PWR(x,y)	$ x ^y$	
PWRS(x,y)	$+ x ^y$ (if $x > 0$) $- x ^y$ (if $x < 0$)	
SIN(x)	$\sin(x)$	where x is in radians
ASIN(x)	$\sin^{-1}(x)$	where the result is in radians
SINH(x)	$\sinh(x)$	where x is in radians
COS(x)	$\cos(x)$	where x is in radians
ACOS(x)	$\cos^{-1}(x)$	where the result is in radians
COSH(x)	$\cosh(x)$	where x is in radians
TAN(x)	$\tan(x)$	where x is in radians
ATAN(x)	$\tan^{-1}(x)$	where the result is in radians
ARCTAN(x)		
ATAN2(y,x)	$\tan^{-1}(y/x)$	where the result is in radians
TANH(x)	$\tanh(x)$	where x is in radians
M(x)	magnitude of x^*	which is the same as ABS(x)
P(x)	phase of x^*	in degrees; returns 0.0 for real numbers
R(x)	real part of x^*	
IMG(x)	imaginary part of x^*	which is applicable to AC analysis only

Table 10 *Functions in arithmetic expressions (continued)*

	This function...	Means this...	
Note <i>In waveform analysis, this function is $D(x)$.</i>	DDT(x)	time derivative of x	which is applicable to transient analysis only
Note <i>In waveform analysis, this function is $S(x)$.</i>	SDT(x)	time integral of x	which is applicable to transient analysis only
	TABLE(x,x ₁ ,y ₁ ,...)	y value as a function of x	where x _n ,y _n point pairs are plotted and connected by straight lines
	MIN(x,y)	minimum of x and y	
	MAX(x,y)	maximum of x and y	
	LIMIT(x,min,max)	min if x < min max if x > max else x	
	SGN(x)	+1 if x > 0 0 if x = 0 -1 if x < 0	
Example: {v(1)*STP(TIME-10ns)} gives a value of 0.0 until 10 nsec has elapsed, then gives v(1).	STP(x)	1 if x > 0 0 otherwise	which is used to suppress a value until a given amount of time has passed
	IF(t,x,y)	x if t is true y otherwise	where t is a relational expression using the relational operators shown in Table 9

* M(x), P(x), R(x), and IMG(x) apply to Laplace expressions only.

Table 11 *System variables*

This variable...	Evaluates to this...
TEMP	<p>Temperature values resulting from a temperature, parametric temperature, or DC temperature sweep analysis.</p> <p>The default temperature, TNOM, is set in the Options dialog box (from the Simulation Settings dialog box, choose the Options tab). TNOM defaults to 27°C.</p> <p>Note <i>TEMP can only be used in expressions pertaining to analog behavioral modelin.</i></p>
TIME	<p>Time values resulting from a transient analysis. If no transient analysis is run, this variable is undefined.</p> <p>Note <i>TIME can only be used in analog behavioral modeling expressions.</i></p>

Note *If a passive or semiconductor device has an independent temperature assignment, then TEMP does not represent that device's temperature.*

To find out more about customizing temperatures for passive or semiconductor devices, refer to the .MODEL command in the *Commands* chapter in the online *OrCAD PSpice A/D Reference Manual*.

Defining power supplies

For the analog portion of your circuit

If the analog portion of your circuit requires DC power, then you need to include a DC source in your design. To specify a DC source, use one of the following parts.

To find out how to use these parts and specify their properties, see the following:

- [Setting up a DC stimulus on page 8-218](#)
- [Using VSRC or ISRC parts on page 3-78](#)

Table 12

For this source type...	Use this part...
voltage	VDC or VSRC
current	IDC or ISRC

Defining stimuli

To simulate your circuit, you need to connect one or more source parts that describe the input signal that the circuit must respond to.

The OrCAD libraries supply several source parts that are described in the tables that follow. These parts depend on:

- the kind of analysis you are running,
- whether you are connecting to the analog portion of your circuit, and
- how you want to define the stimulus: using the Stimulus Editor, using a file specification, or by defining part property values.

Analog stimuli

Analog stimuli include both voltage and current sources. The following table shows the part names for voltage sources.

Table 13

If you want this kind of input...	Use this part for voltage...
For DC analyses	
DC bias	VDC or VSRC
For AC analyses	
AC magnitude and phase	VAC or VSRC
For transient analyses	
exponential	VEXP or VSTIM*
periodic pulse	VPULSE or VSTIM*
piecewise-linear	VPWL or VSTIM*
piecewise-linear that repeats forever	VPWL_RE_FOREVER or VPWL_F_RE_FOREVER**

See [Setting up a DC stimulus on page 8-218](#) for more details.

See [Setting up an AC stimulus on page 9-233](#) for more details.

See [Defining a time-based stimulus on page 10-252](#) for more details.

Table 13

If you want this kind of input...	Use this part for voltage...
piecewise-linear that repeats n times	VPWL_N_TIMES or VPWL_F_N_TIMES**
frequency-modulated sine wave	VSFFM or VSTIM*
sine wave	VSIN or VSTIM*

* VSTIM and ISTIM parts require the Stimulus Editor to define the input signal.

** VPWL_F_RE_FOREVER and VPWL_F_N_TIMES are file-based parts; the stimulus specification is saved in a file and adheres to PSpice netlist syntax.

Example: The current source equivalent to VDC is IDC, to VAC is IAC, to VEXP is IEXP, and so on.

To determine the part name for an equivalent current source

- 1 In the table of voltage source parts, replace the first V in the part name with I.

Using VSTIM and ISTIM

You can use VSTIM and ISTIM parts to define any kind of time-based input signal. To specify the input signal itself, you need to use the Stimulus Editor. See [The Stimulus Editor utility on page 10-253](#).

If you want to specify multiple stimulus types

If you want to run more than one analysis type, including a transient analysis, then you need to use either of the following:

- time-based stimulus parts with AC and DC properties
- VSRC or ISRC parts

Using time-based stimulus parts with AC and DC properties

The time-based stimulus parts that you can use to define a transient, DC, and/or AC input signal are listed below.

VEXP	IEXP
VPULSE	IPULSE
VPWL	IPWL
VPWL_F_RE_FOREVER	IPWL_F_RE_FOREVER
VPWL_F_N_TIMES	IPWL_F_N_TIMES
VPWL_RE_FOREVER	IPWL_RE_FOREVER
VPWL_RE_N_TIMES	IPWL_RE_N_TIMES
VSFFM	ISFFM
VSIN	ISIN

In addition to the transient properties, each of these parts also has a DC and AC property. When you use one of these parts, you must define all of the transient properties. However, it is common to leave DC and/or AC undefined (blank). When you give them a value, the syntax you need to use is as follows.

Table 14

This property...	Has this syntax...
DC	<i>DC_value[units]</i>
AC	<i>magnitude_value[units] [phase_value]</i>

For the meaning of transient source properties, refer to the I/V (independent current and voltage source) device type syntax in the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*.

Using VSRC or ISRC parts

The VSRC and ISRC parts have one property for each analysis type: DC, AC, and TRAN. You can set any or all of them using PSpice netlist syntax. When you give them a value, the syntax you need to use is as follows.

Table 15

This property...	Has this syntax...
DC	<i>DC_value[units]</i>
AC	<i>magnitude_value[units] [phase_value]</i>
TRAN	<i>time-based_type (parameters)</i> where <i>time-based_type</i> is EXP, PULSE, PWL, SFFM, or SIN, and the <i>parameters</i> depend on the <i>time-based_type</i> .

Note *OrCAD recommends that if you are running only a transient analysis, use a VSTIM or ISTIM part if you have the standard package, or one of the other time-based source parts that has properties specific for a waveform shape.*

For the syntax and meaning of transient source specifications, refer to the I/V (independent current and voltage source) device type in the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*.

Things to watch for

This section includes troubleshooting tips for some of the most common reasons your circuit design may not netlist or simulate.

For a roadmap to other commonly encountered problems and solutions, see [When netlisting fails or the simulation does not start on page 3-58.](#)

Unmodeled parts

If you see messages like this in the PSpice Simulation Output window,

```
Warning: Part part_name has no simulation model.
```

then you may have done one of the following things:

- Placed a part from the OrCAD libraries that is not available for simulation (used only for board layout).
- Placed a custom part that has been incompletely defined for simulation.

Do this if the part in question is from the OrCAD libraries

- Replace the part with an equivalent part from one of the libraries listed in the tables below.
- Make sure that you can simulate the part by checking the following:
 - That it has a PSPICETEMPLATE property and that its value is non-blank.
 - That it has an Implementation Type = PSpice MODEL property and that its Implementation property is non-blank.

The libraries listed in the tables that follow all contain parts that you can simulate. Some files also contain parts that you can only use for board layout. That's why you need to check the Pspice TEMPLATE property if you are unsure or still getting warnings when you try to simulate your circuit.

To find out more about a particular library, refer to the online *Library List* or read the header of the model library file itself.

Table 16

Analog libraries with modeled parts (installed in Capture\Library\PSpice)		
1_SHOT	EPWRBJT	NAT_SEMI
ABM	FILTSUB	OPAMP
ADV_LIN	FWBELL	OPTO
AMP	HARRIS	PHIL_BJT
ANALOG	IGBT	PHIL_FET
ANA_SWIT	JBIPOLAR	PHIL_RF
ANLG_DEV	JDIODE	POLYFET
ANL_MISC	JFET	PWRBJT
APEX	JJFET	PWRMOS
BIPOLAR	JOPAMP	SIEMENS
BREAKOUT	JPWRBJT	SWIT_RAV
BUFFER	JPWRMOS	SWIT_REG
BURR_BRN	LIN_TECH	TEX_INST
CD4000	MAGNETIC	THYRISTR
COMLINR	MAXIM	TLINE
DIODE	MOTORAMP	XTAL
EBIPOLAR	MOTORMOS	ZETEX
EDIODE	MOTORSEN	
ELANTEC	MOTOR_RF	

Check for this if the part in question is custom-built

Are there blank (or inappropriate) values for the part's Implementation and PSPICETEMPLATE properties?

If so, load this part into the part editor and set these properties appropriately. One way to approach this is to edit the part that appears in your design.

To edit the properties for the part in question

- 1 In the schematic page editor, select the part.
- 2 From the Edit menu, choose Part.
The part editor window appears with the part already loaded.
- 3 From the Edit menu, choose Properties and proceed to change the property values.

To find out more about setting the simulation properties for parts, see [Defining part properties needed for simulation on page 5-139](#).

To find out more about using the part editor, refer to your *OrCAD Capture User's Guide*.

Unconfigured model, stimulus, or include files

If you see messages like these in the PSpice Simulation Output window,

(*design_name*) Floating pin: *refdes* pin
pin_name

Floating pin: *pin_id*

File not found

Can't open stimulus file

or messages like these in the PSpice output file,

Model *model_name* used by *device_name* is undefined.

Subcircuit *subckt_name* used by *device_name* is undefined.

Can't find .STIMULUS "*refdes*" definition

then you may be missing a model library, stimulus file, or include file from the configuration list, or the configured file is not on the library path.

To find out more about how to configure these files and about search order, see [Configuring model libraries on page 4-120](#).

To find out more about the default configuration, see [How are models organized? on page 4-88](#).

To find out more about the library search path, see [Changing the library search path on page 4-125](#).

Check for this

- Does the relevant model library, stimulus file, or include file appear in the configuration list?
- If the file is configured, does the default library search path include the directory path where the file resides, or explicitly define the directory path in the configuration list?

If the file is not configured, add it to the list and make sure that it appears before any other library or file that has an identically-named definition.

To view the configuration list

- 1 In the Simulation Settings dialog box, click the Include Files tab.

If the directory path is not specified, update the default library search path or change the file entry in the configuration list to include the full path specification.

To view the default library search path

- 1 In the Simulation Settings dialog box, click the Libraries tab.

Unmodeled pins

If you see messages like these in the PSpice Simulation Output window,

Warning: Part *part_name* pin *pin_name* is unmodeled.

Warning: Less than 2 connections at node *node_name*.

or messages like this in the PSpice output file,

Floating/unmodeled pin fixups

then you may have drawn a wire to an unmodeled pin.

The OrCAD libraries include parts that are suitable for both simulation and board layout. The unmodeled pins map into packages but have no electrical significance; PSpice ignores unmodeled pins during simulation.

Check for this

Are there connections to unmodeled pins?

If so, do one of the following:

- Remove wires connected to unmodeled pins.
- If you expect the connection to affect simulation results, find an equivalent part that models the pins in question and draw the connections.

To find out more about searching for parts, see [Finding the part that you want on page 3-62](#).

Missing ground

If for *every net* in your circuit you see this message in the PSpice output file,

```
ERROR -- Node node_name is floating.
```

then your circuit may not be tied to ground.

Check for this

Are there ground parts named 0 (zero) connected appropriately in your design?

If not, place and connect one (or more, as needed) in your design. You can use the 0 (zero) ground part in SOURCE.OLB or any other ground part as long as you change its name to 0.

Missing DC path to ground

If for *selected nets* in your circuit you see this message in the PSpice output file,

```
ERROR -- Node node_name is floating.
```

then you may be missing a DC path to ground.

Check for this

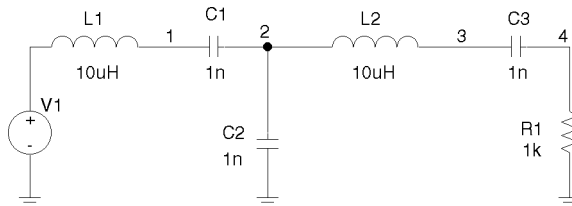
Are there any nets that are isolated from ground by either open circuits or capacitors?

If so, then add a very large (for example, 1 Gohm) resistor either:

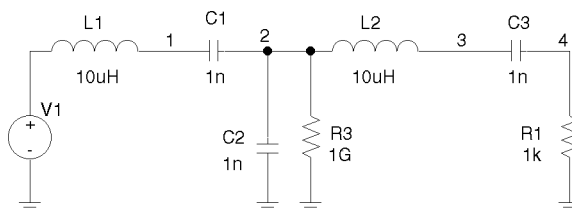
- in parallel with the capacitor or open circuit, or
- from the isolated net to ground.

Note When calculating the bias point solution, PSpice treats capacitors as open circuits and inductors as short circuits.

Example: The circuit shown below connects capacitors (DC open circuits) such that both ends of inductor L2 are isolated from ground.



When simulated, PSpice flags nets 2 and 3 as floating. The following topology solves this problem.



Creating and editing models

4

Chapter overview

This chapter provides information about creating and editing models for parts that you want to simulate.

Topics are grouped into four areas introduced later in this overview. If you want to find out quickly which tools to use to complete a given task and how to start, then:

- 1 Go to the roadmap in [Ways to create and edit models on page 4-92](#).
- 2 Find the task you want to complete.
- 3 Go to the sections referenced for that task for more information about how to proceed.

Background information These sections present model library concepts and an overview of the tools that you can use to create and edit models:

- [What are models? on page 4-87](#)
- [How are models organized? on page 4-88](#)

- [Tools to create and edit models on page 4-91](#)

Task roadmap This section helps you find other sections in this chapter that are relevant to the model editing task that you want to complete:

- [Ways to create and edit models on page 4-92](#)

How to use the tools These sections explain how to use different tools to create and edit models on their own and when editing schematic pages or parts:

- [Using the Model Editor to edit models on page 4-93](#)
- [Editing model text on page 4-110](#)
- [Using the Create Subcircuit command on page 4-115](#)

Other useful information These sections explain how to configure and reuse models after you have created or edited them:

- [Changing the model reference to an existing model definition on page 4-117](#)
- [Reusing instance models on page 4-118](#)
- [Configuring model libraries on page 4-120](#)

What are models?

A model defines the electrical behavior of a part. On a schematic page, this correspondence is defined by a part's Implementation property, which is assigned the model name.

Depending on the device type that it describes, a model is defined as one of the following:

- a model parameter set
- a subcircuit netlist

Both ways of defining a model are text-based, with specific rules of syntax.

Models defined as model parameter sets

PSpice has built-in algorithms or models that describe the behavior of many device types. The behavior of these *built-in models* is described by a *set of model parameters*.

You can define the behavior for a device that is based on a built-in model by setting all or any of the corresponding model parameters to new values using the PSpice **.MODEL** syntax. For example:

```
.MODEL MLOAD NMOS
+ (LEVEL=1 VT0=0.7 CJ=0.02pF)
```

The **.MODEL** syntax applies to the analog models built in to PSpice.

Models defined as subcircuit netlists

For some devices, there are no PSpice built-in models that can describe their behavior fully. These types of devices are defined using the PSpice **.SUBCKT/.ENDS** or *subcircuit syntax* instead.

Subcircuit syntax includes:

- *Netlists* to describe the structure and function of the part.
- *Variable input parameters* to fine-tune the model.

For example:

To find out more about PSpice command and netlist syntax, refer to the online *OrCAD PSpice A/D Reference Manual*.

```
* FIRST ORDER RC STAGE
.SUBCKT LIN/STG IN OUT AGND
+ PARAMS: C1VAL=1 C2VAL=1 R1VAL=1 R2VAL=1
+          GAIN=10000
C1 IN   N1   {C1VAL}
C2 N1   OUT  {C2VAL}
R1 IN   N1   {R1VAL}
R2 N1   OUT  {R2VAL}
EAMP1 OUT AGND VALUE={V(AGND,N1)*GAIN}
.ENDS
```

How are models organized?

The key concepts behind model organization are as follows:

- Model definitions are saved in files called model libraries.
- Model libraries must be configured so that PSpice searches them for definitions.
- Depending on the configuration, model libraries are available either to a specific design or to all (global) designs.

Model libraries

You can use the OrCAD Model Editor, or any standard text editor, to view model definitions in the libraries.

For example: MOTOR_RF.LIB contains models for Motorola-made RF bipolar transistors.

Device model and subcircuit definitions are organized into model libraries. Model libraries are text files that contain one or more model definitions. Typically, model library names have a .LIB extension.

Most model libraries contain models of similar type. For vendor-supplied models, libraries are also partitioned by manufacturer. To find out more about the models contained in a model library, read the comments in the file header.

Model library configuration

PSpice searches model libraries for the model names specified by the MODEL implementation for parts in your design. These are the model definitions that PSpice uses to simulate your circuit.

For PSpice to know where to look for these model definitions, you must configure the libraries. This means:

- Specifying the directory path or paths to the model libraries.
- Naming each model library that PSpice should search and listing them in the needed search order.
- Assigning global or design scope to the model library.

To optimize the search, PSpice uses indexes. To find out more about this and how to add, delete, and rearrange configured libraries, see [Configuring model libraries on page 4-120](#).

Global vs. design models and libraries

Model libraries and the models they contain have either design or global application to your designs.

To find out how to change the design and global configuration of model libraries, see [Changing design and global scope on page 4-123](#).

Design models Design models apply to one design. The *schematic page editor* automatically creates a design model whenever you modify the model definition for a part instance on your schematic page. You can also create models externally and then manually configure the new libraries for a specific design.

Example usage: To set up device and lot tolerances on the model parameters for a particular part instance when running a Monte Carlo or sensitivity/worst-case analysis.

Global models Global models are available to all designs you create. The *part editor* automatically creates a global model whenever you create a part with a new model definition. The Model Editor also creates global models. You can also create models externally and then manually configure the new libraries for use in all designs.

PSpice searches design libraries before global libraries. To find out more, see [Changing model library search order on page 4-124](#).

Nested model libraries

Besides model and subcircuit definitions, model libraries can also contain references to other model libraries using the PSpice .LIB syntax. When searching model libraries for matches, PSpice also scans these referenced libraries.

Example: Suppose you have two custom model libraries, MYDIODES.LIB and MYOPAMPS.LIB, that you want PSpice to search any time you simulate a design. Then you can create a third model library, MYMODELS.LIB, that contains these two statements:

```
.LIB mydiodes.lib  
.LIB myopamps.lib
```

and configure MYMODELS.LIB for global use. Because MYDIODES.LIB and MYOPAMPS.LIB are referenced from MYMODELS.LIB, they are automatically configured for global use as well.

For a list of device models provided by OrCAD, refer to the online *Library List*.

OrCAD-provided models

The model libraries that you initially install with your OrCAD programs are listed in NOM.LIB. This file demonstrates how you can nest references to other libraries and models.

If you click the Libraries tab in the Simulation Settings dialog box immediately after installation, you see the NOM.LIB* entry in the Library Files list. The asterisk means that this model library, and any of the model libraries it references, contain global model definitions.

Tools to create and edit models

There are three tools that you can use to create and edit model definitions. Use the:

- **Model Editor** when you want to:
 - derive models from data sheet curves provided by manufacturers, or
 - modify the behavior of a Model Editor-supported model.
 - edit the PSpice command syntax (text) for .MODEL and .SUBCKT definitions.
- **Create Subcircuit command in the schematic page editor** when you have a hierarchical level in your design that you want to set up as an equivalent part with behavior described as a subcircuit netlist (.SUBCKT syntax).

Note *If you created a subcircuit definition using the Create Subcircuit command and want to alter it, use the Model Editor to edit the definition, or modify the original hierarchical schematic and run Create Subcircuit again to replace the definition.*

For a description of models supported by the Model Editor, see [Model Editor-supported device types on page 4-95](#).

Note *The Create Subcircuit command does not help you create a hierarchical design. You need to create this yourself before using the Create Subcircuit command. For information on hierarchical designs and how to create them, refer to the OrCAD Capture User's Guide.*

Ways to create and edit models

This section is a roadmap to other information in this chapter. Find the task that you want to complete, then go to the referenced sections for more information.

If you want to...	Then do this...	To find out more, see this...
➔ Create or edit the model for an existing part and have it affect all designs that use that part.	Create or load the part first in the part editor, then edit the model using the Model Editor *.	Running the Model Editor from the schematic page editor on page 4-101.
➔ Create a model from scratch and automatically create a part for it to use in any design.	Start the Model Editor * and enable/disable automatic part creation as needed; then create or view the model.	Running the Model Editor alone on page 4-99.
➔ Create a model from scratch without a part and have the model definition available to any design.		
➔ View model characteristics for a part.		
➔ Define tolerances on model parameters for statistical analyses.	Select the part instance on your schematic, then edit the model using the Model Editor.	Starting the Model Editor from the schematic page editor in Capture on page 4-111.
➔ Test behavior variations on a part.	Select the part instance on your schematic page, then edit the model using the Model Editor *.	Running the Model Editor from the schematic page editor on page 4-101
➔ Refine a model before making it available to all designs.		Starting the Model Editor from the schematic page editor in Capture on page 4-111.
➔ Derive subcircuit definitions from a hierarchical design.	Use the Create Subcircuit command in the schematic page editor.	Using the Create Subcircuit command on page 4-115.

* For a list of device types that the Model Editor supports, see [Model Editor-supported device types on page 4-95](#). If the Model Editor does not support the device type for the model definition that you want to create, then you can edit the text using the Model Editor to create a model definition using the PSpice .MODEL and .SUBCKT command syntax. Remember to configure the new model library.

Using the Model Editor to edit models

The Model Editor converts information that you enter from the device manufacturer's data sheet into either:

- model parameter sets using PSpice .MODEL syntax, or
 - subcircuit netlists using PSpice .SUBCKT syntax,
- and saves these definitions to model libraries that PSpice can search when looking for simulation models.

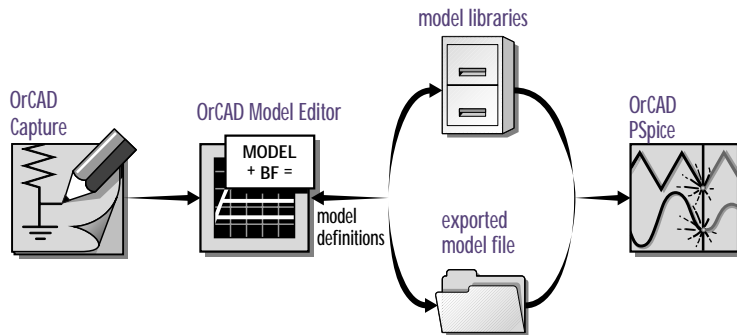


Figure 27 Relationship of the Model Editor to Capture and PSpice.

Note By default, the Model Editor creates or updates model libraries. To create an exported model file, choose the Export command from the Model menu and configure it as an include file. For more information, see [How PSpice uses model libraries](#) and the companion sidebar on page [4-121](#).

The Normal view in the Model Editor does not support the following subcircuit constructs:

- optional nodes construct, OPTIONAL:
- variable parameters construct, PARAMS:
- local .PARAM command
- local .FUNC command

To refine the subcircuit definition for these constructs, use the Model Text view in Model Editor, described in [Editing model text on page 4-110](#).

Ways to use the Model Editor

You can use the Model Editor five ways:

To find out more, see [Running the Model Editor alone on page 4-99.](#)

To find out more, see [Running the Model Editor alone on page 4-99.](#)

To find out more, see [Running the Model Editor from the schematic page editor on page 4-101.](#)

To find out more, see [Running the Model Editor alone on page 4-99.](#)

- **To define a new model, and then automatically create a part.** Any new models and parts are automatically available to any design.
- **To define a new model only (no part).** You can optionally turn off the part creation feature for new models. The model definition is available to any design, for example, by changing the model implementation for a part instance.
- **To edit a model definition for a part instance on your schematic.** This means you need to start the Model Editor from the schematic page editor after selecting a part instance on your schematic. The schematic editor automatically attaches the new model implementation (that the Model Editor creates) to the selected part instance.
- **To examine or verify the electrical characteristics of a model without running PSpice.** This means you can use the Model Editor alone to:
 - check characteristics of a model quickly, given a set of model parameter values, or
 - compare characteristic curves to data sheet information or measured data.

Model Editor-supported device types

Table 17 summarizes the device types supported in the Model Editor.

Table 17 *Models supported in the Model Editor*

This part type...	Uses this definition form...	And this name prefix* ...
diode	.MODEL	D
bipolar transistor	.MODEL	Q
bipolar transistor, Darlington model	.SUBCKT	X
IGBT	.MODEL	Z
JFET	.MODEL	J
power MOSFET	.MODEL	M
operational amplifier**	.SUBCKT	X
voltage comparator**	.SUBCKT	X
nonlinear magnetic core	.MODEL	K
voltage regulator**	.SUBCKT	X
voltage reference**	.SUBCKT	X

* This is the standard PSpice device letter notation. Refer to the online *OrCAD PSpice A/D Reference Manual*.

** The Model Editor only supports .SUBCKT models that were generated by the Model Editor. However, you can edit the text of a .SUBCKT model created manually, or by another tool, using the Model Editor. When you load a .SUBCKT model that the Model Editor did not create, the Model Editor displays the text of the model for editing.

Device types that the Model Editor models using the .MODEL statement are based on the models built into PSpice.

Note *The model parameter defaults used by the Model Editor are different from those used by the models built into PSpice.*

Testing and verifying models created with the Model Editor

Each curve in the Model Editor is defined only by the parameters being adjusted. For the diode, the forward current curve *only* shows the part of the current equation that is associated with the forward characteristic parameters (such as I_S , N , R_s).

However, PSpice uses the *full* equation for the diode model, which includes a term involving the reverse characteristic parameters (such as I_{SR} , NR). These parameters could have a significant effect at low current.

This means that the curve displayed in the Model Editor does not exactly match what is displayed in PSpice after a simulation. Be sure to test and verify models using PSpice. If needed, fine-tune the models.

Ways To Characterize Models

Figure 28 shows two ways to characterize models using the Model Editor.

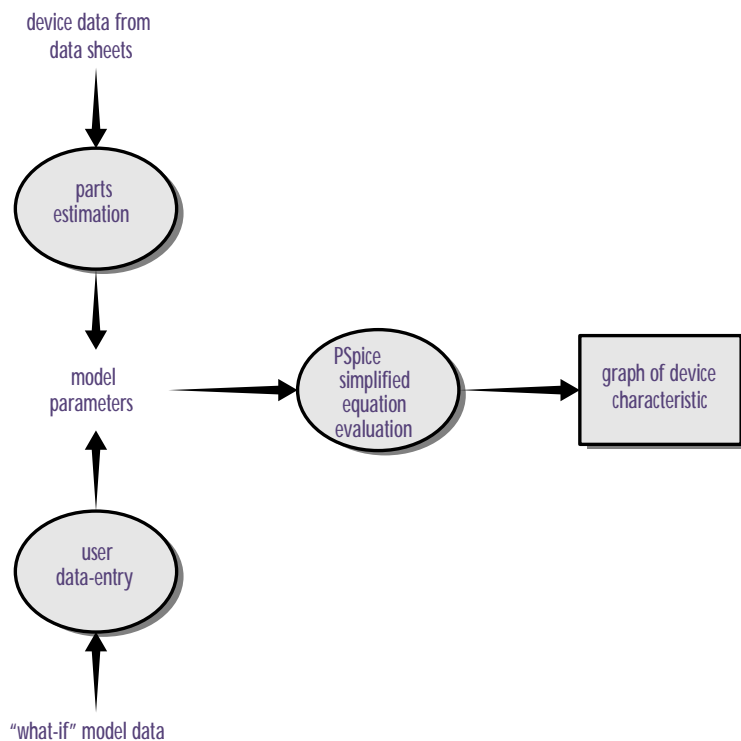


Figure 28 Process and data flow for the Model Editor.

Note When specifying operating characteristics for a model, you can use typical values found on data sheets effectively for most simulations. To verify your design, you may also want to use best- and worst-case values to create separate models, and then swap them into the circuit design.

Creating models from data sheet information

The most common way to characterize models is to enter data sheet information for each device characteristic. After you are satisfied with the behavior of each characteristic, you can have the Model Editor estimate (or *extract*) the corresponding model parameters and generate a graph showing the behavior of the characteristic. This is called the fitting process.

You can repeat this process, and when you are satisfied with the results, save them; the Model Editor creates

model libraries containing appropriate model and subcircuit definitions.

Analyzing the effect of model parameters on device characteristics

You can also edit model parameters directly and see how changing their values affects a device characteristic. As you change model parameters, the Model Editor recalculates the behavior of the device characteristics and displays a new curve for each of the affected ones.

How to fit models

For a given model, the Model Editor displays a list of the device characteristics and a list of all model parameters and performance curves (see Figure 29).

For more information about the characteristics of devices supported by the Model Editor, refer to the online *OrCAD PSpice A/D Reference Manual*.

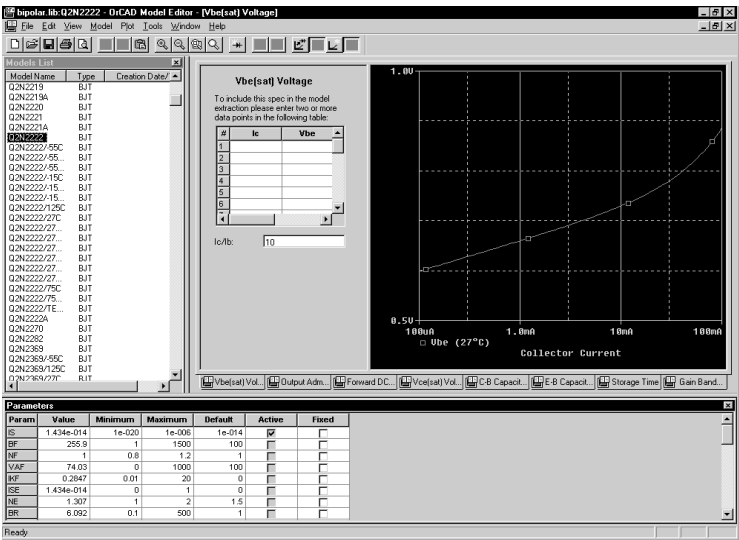


Figure 29 Model Editor workspace with data for a bipolar transistor.

To fit the model

- 1 For each device characteristic that you want to set up:
 - a In the Spec Entry frame, click the tab of the device characteristic.
 - b Enter the device information from the data sheet.



- 2 From the Tools menu, choose Extract Parameters to extract all relevant model parameters for the current specification.

A check mark appears in the Active column of the Parameters frame for each extracted model parameter.

- 3 Repeat steps [1-2](#) until the model meets target behaviors.

To view updated performance curves



- 1 On the toolbar, click the Update Graph button.

Note *If you view performance curves before fitting, then your data points and the curve for the current model specification may not match.*

Running the Model Editor alone

Run the Model Editor alone if you want to do any of the following:

- create a model and use the model in any design (and automatically create a part),
- create a model and have the model definition available to any design (without creating a part), or
- examine or verify the characteristics of a given model without using PSpice.

Running the Model Editor alone means that the model you are creating or examining is not currently tied to a part instance on your schematic page or to a part editing session.

Note *You can only edit models for device types that the Model Editor supports. See [Model Editor-supported device types on page 4-95](#) for details.*

After you have selected the part that you want to model, you can proceed with entering data sheet information and model fitting as described in [How to fit models on page 4-97](#).

Starting the Model Editor

To start the Model Editor alone

- 1 From the Start menu, point to the OrCAD program folder, then choose Model Editor.
- 2 From the File menu, choose New or Open, and enter an existing or new model library name.
- 3 From the Part menu, choose New, Copy From, or Import to load a model.

If you have already started the Model Editor from Capture and want to continue working on new models, then:

- 1 Save the opened model library.
- 2 Open or create a different model library.
- 3 Get a model, or create a new one.

Instead of using the OrCAD default part set for new models, you can have the Model Editor use your own set of standard parts. To find out more, see [Basing new parts on a custom set of parts on page 5-133](#).

Example: If the model library is MYPARTS.LIB, then the Model Editor creates the part library MYPARTS.OLB.

If you want to save the open model library to a new library, then:

- 1 From the File menu, choose Save As.
- 2 Enter the name of the new model library.

If you want to save only the model definition that you are currently editing to a different library, then

- 1 From the Part menu, select Export.
- 2 Enter the name of the new file.
- 3 If you want PSpice to search this file automatically, configure it in Capture (using the Libraries tab on the Simulation Settings dialog box).

Enabling and disabling automatic part creation

Part creation in the Model Editor is optional. By default, automatic part creation is enabled. However, if you previously disabled part creation, you will need to enable it before creating a new model and part.

To automatically create parts for new models

- 1 From the Tools menu, choose Options.
- 2 If not already checked, select Always Create Part to enable automatic part creation.
- 3 Under Save Part To, enter the name of the part library for the new part. Choose either:
 - Part Library Path Same As Model Library to create or open the *.OLB file that has the same name prefix as the currently open model library (*.LIB).
 - User-Defined Part Library, and then enter a file name in the Part Library Name text box.

Note *If you select a user-defined Part library, the Model Editor saves all new parts to the specified file until you change it.*

Saving global models (and parts)

When you save your changes, the Model Editor does the following for you:

- Saves the model definition to the model library that you originally opened.
- If you had the automatic part creation option enabled, saves the part definition to MODEL_LIBRARY_NAME.OLB.

To save the new model (and part)

- 1 From the File menu, choose Save to update MODEL_LIBRARY_NAME.LIB (and, if you enabled part creation, MODEL_LIBRARY_NAME.OLB), and save them to disk.

Running the Model Editor from the schematic page editor

If you want to:

- test behavior variations on a part, or
- refine a model before making it available to all designs,

then run the Model Editor from the schematic page editor in Capture.

This means editing models for part instances on your schematic page. When you select a part instance and edit its model, the schematic page editor automatically creates an *instance model* that you can then change.

Note You can only edit models for device types that the Model Editor supports. See [Model Editor-supported device types on page 4-95](#) for details.

What is an instance model?

An instance model is a *copy* of the part's original model. The copied model is local to the design. You can customize the instance model without impacting any other design that uses the original part from the library.

When the schematic editor creates the copy, it assigns a unique name that is by default:

original_model_name-Xn

where *n* is <blank 1 | 2 | ... > depending on the number of different instance models derived from the original model for the current design.

Once you have started the Model Editor, you can proceed with entering data sheet information and model fitting as described in [How to fit models on page 4-97](#).

For more information on instance models, see [Reusing instance models on page 4-118](#).

Starting the Model Editor

To start editing an instance model

- 1 In Capture, select one part on your schematic page.
- 2 From the Edit menu, choose PSpice Model.

The schematic page editor searches the model libraries for the instance model.

- If found, the schematic page editor starts the Model Editor, which opens the model library that contains the instance model and loads the instance model.
- If not found, the schematic page editor assumes that this is a new instance model and does the following: makes a copy of the original model definition, names it *original_model_name-Xn*, and starts the Model Editor with the new model loaded.

To find out how Capture searches the library, see [Changing model library search order on page 4-124](#).

Saving design models

When you save your edits, the Model Editor saves the model definition to *DESIGN_NAME.LIB*, which is already configured for local use (see [What happens if you don't save the instance model on page 4-103](#)).

To save instance models

- 1 From the File menu, choose Save to update *DESIGN_NAME.LIB* and save it to disk.

What happens if you don't save the instance model

Before the schematic page editor starts the Model Editor, it does these things:

- Makes a copy of the original model and saves it as an instance model in *SCHEMATIC_NAME.LIB*.
- Configures *SCHEMATIC_NAME.LIB* for design use, if not already done.
- Attaches the new instance model name to the Implementation property for the selected part instance.

This means that if you:

- quit the Model Editor, or
- return to Capture to simulate the design

without first saving the model you are editing, the part instance on your schematic page is still attached to the instance model implementation.

In this case, the instance model is identical to the original model. If you decide to edit this model later, be sure to do one of the following:

- If you want the changes to remain specific to the current design, edit the instance model in the design library, using the Model Editor.
- If you want the change to be global, change the model implementation for the part instance in your design back to the original model name in the global library, and then edit the original model from within the part editor.

To find out how to change model references, see [Changing the model reference to an existing model definition on page 4-117](#).

The Model Editor tutorial

In this tutorial, you will model a simple diode device as follows:

- Create the schematic for a simple half-wave rectifier.
- Run the Model Editor from the schematic editor to create an instance model for the diode in your schematic.

Creating the half-wave rectifier design

To draw the design

- 1 From the Project Manager, from the File menu point to New, then choose Project.
- 2 Enter the name of the new project (RECTFR) and click Create.
- 3 From Capture's Place menu, choose Part.
- 4 Place one each of the following parts (reference designator shown in parentheses) as shown in Figure 30:
 - Dbreak (D1 diode)
 - C (C1 capacitor)
 - R (R1 resistor)
 - VSIN (V1 sine wave source)
- 5 Click the Ground button and place the analog ground.
- 6 From the Place menu, choose Wire, and draw the connections between parts as shown in Figure 30.
- 7 From the File menu, choose Save.

Note If you were to simulate this design using a transient analysis, you would also need to set up a transient specification for V1; most likely, this would mean defining the VOFF (offset voltage), VAMPL (amplitude), and FREQ (frequency) properties for V1. For this tutorial, however, you will not perform a simulation, so you can skip this step.

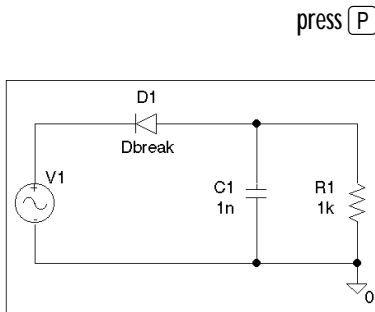


Figure 30 Design for a half-wave rectifier.

press **W**

Using the Model Editor to edit the D1 diode model

To create a new model and model library

- 1 In the Model Editor, from the Model menu, choose New.
- 2 In the New dialog box, do the following:
 - a In the Model text box, type DbreakX.
 - b From the From Model list, select Diode.
 - c Click OK.
- 3 From the File menu, choose Save As.
- 4 In the File name text box, type `rectfr.lib` to save the library as RECTFR.LIB.



Entering data sheet information

As shown in Figure 31, the Model Editor initially displays:

- diode model characteristics listed in the Models List frame, and
- DbreakX model parameter values listed in the Parameters frame.

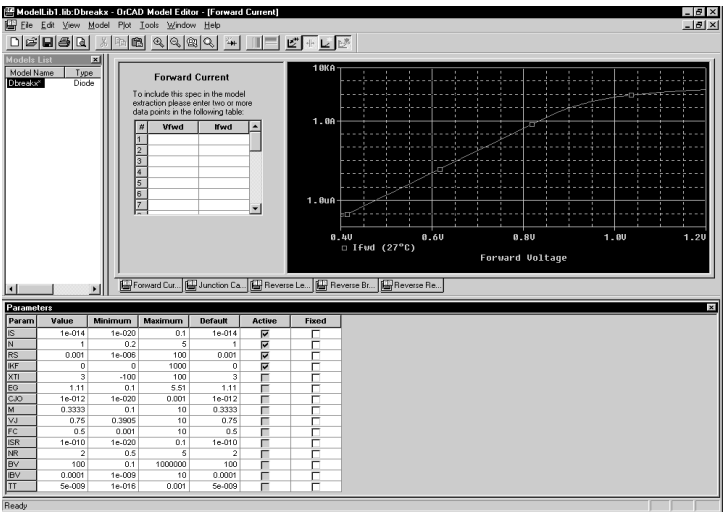


Figure 31 Model characteristics and parameter values for DbreakX.

You can modify each model characteristic shown in the Model Spec frame with new values from the data sheets. The Model Editor takes the new information and fits new model parameter values.

When updating the entered data, the Model Editor expects either:

- device curve data (point pairs), or
- single-valued data,

depending on the device characteristic.

For the diode, Forward Current, Junction Capacitance, and Reverse Leakage require device curve data. Reverse Breakdown and Reverse Recovery require single-valued data.

Table 1 lists the data sheet information for the Dbreak-X model.

Table 1 *Sample diode data sheet values*

For this model characteristic...	Enter this...
forward current	(1.3, 0.2)
junction capacitance	(1m, 120p) (1, 73p) (3.75, 45p)
reverse leakage	(6, 20n)
reverse breakdown	(Vz=7.5, Iz=20m, Zz=5)
reverse recovery	no changes

To change the Forward Current characteristic

- 1 In the Spec Entry frame, click the Forward Current tab.
This tab requires curve data.
- 2 In the Vfwd text box, type 1 . 3.
- 3 Press **Tab** to move to the Ifwd text box, and then type 0 . 2.

To change the values for Junction Capacitance and Reverse Leakage

- 1 Follow the same steps as for Forward Current, entering the data sheet information listed in [Table 1](#) that corresponds to the current model characteristic.

To change the Reverse Breakdown characteristic

- 1 In the Spec Editing frame, click the Reverse Breakdown tab.

This tab requires single-valued data.

- 2 In the Vz text box, type 7.5.
- 3 Press to move to the Iz text box, and then type 20m.
- 4 Press to move to the Zz text box, and then type 5.

The Model Editor accepts the same scale factors normally accepted by PSpice.

Extracting model parameters

To generate new model parameter values

- 1 From the Tools menu, choose Extract Parameters.
A check mark appears in the Active column of the Parameters frame for each extracted model parameter.



To display the curves for the five diode characteristics

- 1 From the Window menu, choose Tile.
Some of the plots are shown in Figure 32 below.

You can also do the following with an active plot window:

- Pan and zoom within the plot using commands on the View menu.
- Rescale axes using the Axis Settings command on the Plot menu.

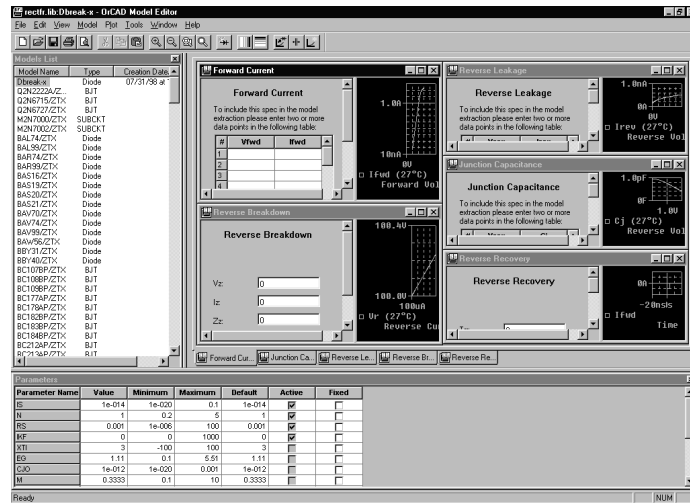


Figure 32 Assorted device characteristic curves for a diode.

Adding curves for more than one temperature

By default, the Model Editor computes device curves at 27°C. For any characteristic, you can add curves to the plot at other temperatures.

To add curves for Forward Current at a different temperature

- 1 In the Spec Entry frame, click the Forward Current tab.
- 2 From the Plot menu, choose Add Trace.
- 3 Type 100 (in °C).
- 4 Click OK.

The Forward Current plot should appear as shown in Figure 33 below.

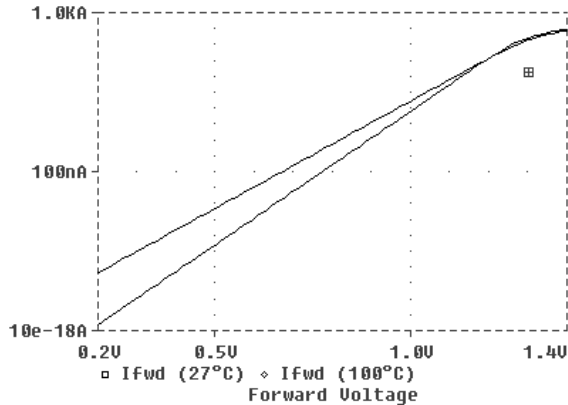


Figure 33 *Forward Current device curve at two temperatures.*

Completing the model definition

You can refine the model definition by:

- modifying the entered data as described before, or
- editing model parameters directly.

You can update individual model parameters by editing them in the Parameters frame of the Model Editor workspace. When you save the model library, the Model Editor automatically updates the device curves.

For this tutorial, leave the model parameters at their current settings.

To save the model definition with the current parameter values and to make the model available to your design

- 1 From the File menu, select Save to update RECTFR.LIB and save the library to disk.

Your design is ready to simulate with the model definition you just created.

Editing model text

Caution—If you edit the text of a model that was created by entering data sheet values, you may not be able to edit the model in Normal view again.

For any model, you can edit model text in the Model Editor instead of using the Spec Entry and Parameter frames. However, there are two cases where you must edit the model text:

- When you want to edit models of device types not supported by the Model Editor. The model text is displayed automatically when you load one of these models.
- When you want to add DEV and LOT tolerances to a model for Monte Carlo or sensitivity/worst-case analysis.

By typing PSpice commands and netlist entries, you can do the following:

- change definitions, and
- create new definitions

When you are finished, the Model Editor automatically configures the model definitions into the model libraries.

To display the model text

- 1 From the View menu, choose Model Text.

The Model Editor displays the PSpice syntax for model definitions:

- .MODEL syntax for models defined as parameter sets
- .SUBCKT syntax for models defined as netlist subcircuits

You can edit the definition just as you would in any standard text editor.

Editing .MODEL definitions

For definitions implemented as model parameter sets using PSpice .MODEL syntax, the Model Editor lists one parameter per line. This makes it easier to add DEV/LOT

To find out more about PSpice command and netlist syntax, refer to the online *OrCAD PSpice A/D Reference Manual*.

tolerances to model parameters for Monte Carlo or sensitivity/worst-case analysis.

Editing .SUBCKT definitions

For definitions implemented as subcircuit netlists using PSpice .SUBCKT syntax, the model editor displays the subcircuit syntax exactly as it appears in the model library. The Model Editor also includes all of the comments immediately before or after the subcircuit definition.

Changing the model name

You can change the model name directly in the PSpice .MODEL or .SUBCKT syntax, but double-check that the new name does not conflict with models already contained in the libraries.

Note *If you do create a model with the same name as another model and want PSpice to always use your model, make sure the configured model libraries are ordered so your definition precedes any other definitions.*

To find out more about instance model naming conventions, see [What is an instance model? on page 4-112](#).

To find out more about search order in the model library, see [Changing model library search order on page 4-124](#).

Starting the Model Editor from the schematic page editor in Capture

Start the model editor from the schematic page editor in Capture when you want to:

- define tolerances on model parameters for statistical analyses,
- test behavior variations on a part, or
- refine a model before making it available to all designs.

This means editing models for part instances in your design. When you select a part instance and edit its model, the schematic page editor automatically creates an *instance model* that you can then change.

You can also use the model editor to view the syntax for a model definition. When you are finished viewing, be sure to quit the Model Editor without saving the library, so the schematic page editor does not create an instance model.

For more information on instance models, see [Reusing instance models on page 4-118](#).

After you start the Model Editor, you can proceed to change the text as described in [To display the model text on page 4-110](#).

To find out how Capture searches the library, see [Changing model library search order on page 4-124](#).

What is an instance model?

An instance model is a *copy* of the part's original model. The copied model is limited to use in the current design. You can customize the instance model without impacting any other design that uses the original part from the library.

When the schematic page editor creates the copy, it assigns a unique name that is by default:

original_model_name-Xn

where *n* is *<blank 1 | 2 | ... >* depending on the number of different instance models derived from the original model for the current design.

Starting the Model Editor

To start editing an instance model

- 1 In the schematic page editor, select the part on the schematic page.
- 2 From the Edit menu, choose PSpice Model.

The schematic page editor searches the configured libraries for the instance model:

- If found, the schematic page editor starts the Model Editor, which opens the library containing the instance model and displays the model for editing.
- If not found, the schematic page editor assumes that this is a new instance model and starts the Model Editor, which does the following: makes a copy of the original model definition, names it *original_model_name-Xn*, and displays the new model text for editing.

Saving design models

When you save your edits, the following is done for you to make sure the instance model is linked to the selected part instances in your design:

- The Model Editor saves the model definition to *DESIGN_NAME.LIB*.
- If the library is new, the Model Editor configures *DESIGN_NAME.LIB* for local use.
- The schematic page editor assigns the new model name to the Implementation property for each of the selected part instances.

To save instance models

- 1 In the Model Editor, from the File menu, choose Save.
- 2 From the File menu, choose Exit to quit the Model Editor.

Actions that automatically configure the instance model library for global use instead

Instance model libraries are normally configured for design use. However, if you perform the following action, the model editor configures the library for global use instead:

- Save the model to a different library by typing a new file name in the Library text box in the Save To frame.

Example: editing a Q2N2222 instance model

Suppose you have a design named MY.OPJ that contains several instances of a Q2N2222 bipolar transistor.

Suppose also that you are interested in the effect of base resistance variation on one specific device: Q6. To do this, you need to do the following:

- Define a tolerance (in this example, 5%) on the Rb model parameter.
- Set up and run a Monte Carlo analysis.

The following example demonstrates how to set up the instance model for Q6.

Starting the Model Editor

To start the Model Editor, you need to:

- 1 In the schematic page editor, select Q6 on the schematic page.
- 2 From the Edit menu, choose PSpice Model.

The Model Editor automatically creates a copy of the Q2N2222 base model definition.

- 3 In the Model Editor, from the View menu, choose Model Text.

The Model Editor displays the PSpice syntax for the copied model in the text editing area.

Editing the Q2N2222-X model instance

Text edits appropriate to this example are as follows:

- Add the DEV 5% clause to the Rb statement (required).
- Change the model name to Q2N2222-MC (optional, for descriptive purposes only).

To find out more about PSpice command and netlist syntax, refer to the online *OrCAD PSpice A/D Reference Manual*.

Saving the edits and updating the schematic

When you choose Save from the File menu, two things happen:

- The Model Editor saves the model definition to the model library.
- The schematic page editor updates the Implementation property value to Q2N2222-MC for the Q6 part instance.

In this example, the default model library is MY.LIB. If MY.LIB does not already exist, the Model Editor creates and saves it in the current working directory. The schematic page editor then automatically configures it as a design model library for use with the current design only.

Now you are ready to set up and run the Monte Carlo analysis.

If you verify the model library configuration (in the Simulation Settings dialog box, click the Libraries tab), you see entries for NOM.LIB* (for global use, as denoted by the asterisk) and MY.LIB (for design use, no asterisk) in the Library files list.

You can change the model reference for this part back to the original Q2N2222 by following the procedure [To change model references for part instances on your design on page 4-117](#).

Using the Create Subcircuit command

The Create Subcircuit command creates a subcircuit netlist definition for the displayed level of hierarchy and all lower levels in your design.

The schematic page editor does the following things for you:

- Maps any named interface ports at the active level of hierarchy to terminal nodes in the PSpice .SUBCKT statement.
- Saves the subcircuit definition to a file named *DESIGN_NAME.SUB*.

Before you can use the subcircuit definition in your design, you need to:

- Create a part for the subcircuit.

The Create Subcircuit command does not help you create a hierarchical design. You need to do this yourself before using the Create Subcircuit command. For information on hierarchical designs and how to create them, refer to the *OrCAD Capture User's Guide*.

- Configure the *DESIGN_NAME.SUB* file so PSpice knows where to find it.

To create a subcircuit definition for a portion of your design

To create a part for the subcircuit

- 1 In the schematic page editor, move to the level of hierarchy for which you want to create a subcircuit (.SUBCKT) definition.
- 2 From the Place menu, choose Hierarchical Port.
- 3 From the File menu, choose Save.
- 4 In the Project Manager, from the Tools menu, choose Create Netlist.
- 5 Select the PSpice tab.
- 6 In the Options frame, select Create SubCircuit Format Netlist.
- 7 Click OK to generate the subcircuit definition and save it to *DESIGN_NAME.SUB*.

To configure the subcircuit file

- 1 In the schematic page editor, from the PSpice menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.
- 2 Click either the Libraries tab or the Include Files tab, then configure *DESIGN_NAME.SUB* as either a model library or an include file (see [Configuring model libraries on page 4-120](#)).
- 3 If necessary, refine the subcircuit definition for the new part or for a part instance on your schematic page using the Model Editor (see [Editing model text on page 4-110](#)).
- 4 From Capture's Edit menu, choose Part to start the part editor.
- 5 Create a new part for the subcircuit definition.

Refinements can include extending the subcircuit definition using the optional nodes construct, OPTIONAL:, the variable parameters construct, PARAMS:, and the .FUNC and local .PARAM commands.

One way to do this is to use the part wizard. See [Chapter 5, Creating parts for models](#) for a complete discussion.

Changing the model reference to an existing model definition

Parts are linked to models by the model name assigned to the parts' Implementation property. You can change this assignment by replacing the Implementation property value with the name of a different model that already exists in the library.

You can do this for:

- A part instance in your design.
- A part in the part library.

To change model references for part instances on your design

- 1 Find the name of the model that you want to use.
- 2 In the schematic page editor, select one or more parts on your schematic page.
- 3 From the Edit menu, choose Properties.
The Parts spreadsheet appears.
- 4 Click the cell under the column Implementation Type.
- 5 From the Implementation list, select PSpice Model.
- 6 In the Implementation column, type the name of the existing model that you want to use if it is not already listed.
- 7 Click Apply to update the changes, then close the spreadsheet.

To change the model reference for a part in the part library

- 1 Find the name of the model that you want to use.
- 2 In the schematic page editor, select the part you want to change.
- 3 From the Edit menu, choose Part to start the part editor with that part loaded for editing.

- 4 From the Options menu, choose Part Properties to display the User Properties dialog box.
- 5 Select Implementation Type.
- 6 From the Implementation list, select PSpice Model.
- 7 In the Implementation text box, type the name of the existing model that you want to use if it is not already listed.
- 8 Click OK to close the Edit Part dialog box.

Reusing instance models

For information on how to create instance models, see:

- [Running the Model Editor from the schematic page editor on page 4-101.](#)
- [Starting the Model Editor from the schematic page editor in Capture on page 4-111.](#)

If you created instance models in your design and want to reuse them, there are two things you can do:

- Attach the instance model implementation to other part instances in the same design.
- Change the instance model to a global model and create a part that corresponds to it.

Reusing instance models in the same schematic

There are two ways to use the instance model elsewhere in the same design.

To use the instance model elsewhere in your design

- 1 Do one of the following:
 - Change the model reference for other part instances to the name of the new model instance.
 - From the Edit menu, use the Copy and Paste commands to place more part instances.

See [Changing the model reference to an existing model definition on page 4-117.](#)

Making instance models available to all designs

If you are refining model behavior specific to your design, and are ready to make it available to any design, then you need to link the model definition to a part and configure it for global use.

To make your instance model available to any design

- 1 Create a part and assign the instance model name to the Implementation property.
- 2 If needed, move the instance model definition to an appropriate model library, and make sure the library is configured for global use.

Note *If you use the part wizard to create the part automatically from the model definition, then this step is completed for you.*

See [Chapter 5, Creating parts for models](#) for more information.

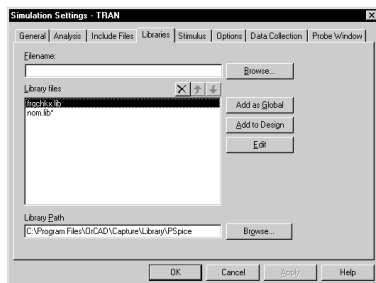
See [Configuring model libraries on page 4-120](#) for more information.

Configuring model libraries

Although model libraries are usually configured for you, there are things that you sometimes must do yourself.

These are:

- adding new model libraries that were created outside of Capture or the Model Editor
- changing the global or design scope of a model library
- changing the library search order
- changing or adding directory search paths



The Libraries and Include Files tabs

The Libraries and Include Files tabs of the Simulation Settings dialog box are where you can add, change, and remove model libraries and include files from the configuration or resequence the search order.

Note *Removing a library in this dialog box means that you are removing the model library from the configured list. The library still exists on your computer and you can add it back to the configuration later.*

To display the Libraries tab

- 1 In PSpice, from the Simulation menu, choose Edit Simulation Settings.
- 2 Click the Libraries tab.

The Library Files list shows the model libraries that PSpice searches for definitions matching the parts in your design. Files showing an asterisk (*) after their name have global scope; files with names left unmarked have design scope.

The buttons for adding model libraries to the configuration follow the same local/global syntax convention. Click one of the following:

- Add to Design for design models.

The Include Files tab contains include files. You can manually add design and global include files to your configuration using the Add to Design and Add as Global buttons, respectively.

The Stimulus tab contains stimulus files. See [Configuring stimulus files on page 10-254](#) for more information.

- Add as Global for global models.

How PSpice uses model libraries

PSpice searches libraries for any information it needs to complete the definition of a part or to run a simulation. If an up-to-date index does not already exist, PSpice automatically generates an index file and uses the index to access only the model definitions relevant to the simulation. This means:

- Disk space is not used up with definitions that your design does not use.
- There is no memory penalty for having large model libraries.
- Loading time is kept to a minimum.

Search order

When searching for model definitions, PSpice scans the model libraries using these criteria:

- design model libraries before global model libraries
- model library sequence as listed in the Libraries tab of the Simulation Settings dialog box
- local directory (where the current design resides) first, then the list of directories specified in the library search path in the order given (see [Changing the library search path on page 4-125](#))

Caution—When you use include files instead

PSpice treats model library and include files differently as follows:

- For model library files, PSpice reads in only the definitions it needs to run the current simulation.
- For include files, PSpice reads in the file in its entirety.

This means if you configure a model library (*.LIB extension) as an include file using the Add to Design or Add as Global button, PSpice loads every model definition contained in that file.

If the model library is large, you may overload the memory capacity of your system. However, when developing models, you can do the following:

- 1 Initially configure the model library as an include file; this avoids rebuilding the index files every time the model library changes.
- 2 When your models are stable, reconfigure the include file containing the model definitions as a library file.

To reconfigure an include file as a library file:

- 1 From the Simulation menu, choose Edit Simulation Settings, then click the Include Files tab.
- 2 Select the include file that you want to change.
- 3 Click either the Add as Global or the Add to Design button.
- 4 Click Remove to remove the include file entry.

Handling duplicate model names

If your model libraries contain duplicate model names, PSpice always uses the first model it finds. This means you might need to resequence the search order to make sure PSpice uses the model that you want. See [Changing model library search order on page 4-124](#).

Note *PSpice searches design libraries before global libraries, so if the new model you want to use is specific to your design and the duplicate definition is global, you do not need to make any changes.*

Adding model libraries to the configuration

New libraries are added above the selected library name in the Library Files list box.

To add model libraries to the configuration

- 1 From the Simulation menu, choose Edit Simulation Settings, then click the Libraries tab.
- 2 Click the library name positioned one entry below where you want to add the new library.
- 3 In the Filename text box, either:
 - type the name of the model library, or
 - click Browse to locate and select the library.
- 4 Do one of the following:
 - If the model definitions are for use in the current design only, click the Add to Design button.
 - If the model definitions are for global use in any schematic, click the Add as Global button instead.
- 5 Click OK.

Note *If the model libraries reside in a directory that is not on the library search path, and you use the Browse button in step 3 to select the libraries you want to add, then the schematic editor automatically updates the library search path. Otherwise, you need to add the directory path yourself. See [Changing the library search path on page 4-125](#).*

Changing design and global scope

There are times when you might need to change the scope of a model library from design to global, or vice versa.

To change the scope of a design model to global

- 1 From the Simulation menu, choose Edit Simulation Settings, then click the Libraries tab.
- 2 Select the model library that you want to change.
- 3 Do one of the following:
 - Click the Add as Global button to add a global entry.
 - Click the Add to Design button to add a design entry.
- 4 Click the Delete toolbar button to remove the local entry.

Example: If you have an instance model that you now want to make available to any design, then you need to change the local model library that contains it to have global scope.

For more information, see [Global vs. design models and libraries on page 4-89](#).

Changing model library search order

Two reasons why you might want to change the search order are to:

- reduce the search time
- avoid using the wrong model when there are model names duplicated across libraries; PSpice always uses the first instance

See [Handling duplicate model names on page 4-122](#) for more information.

To change the order of libraries

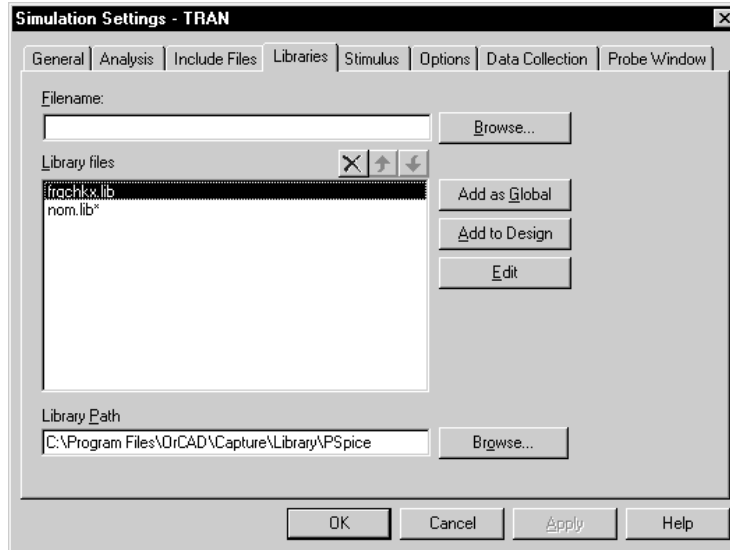
- 1 On the Libraries tab of the Simulation Settings dialog box:
 - a Select the library name you wish to move.
 - b Use either the Up Arrow or Down Arrow toolbar button to move the library name to a different place in the list.
- 2 If you have listed multiple *.LIB commands within a single library (like NOM.LIB), then edit the library using a text editor to change the order.

Example: The model libraries DIODES.LIB and EDIODES.LIB (European manufactured diodes) shipped with your OrCAD programs have identically named device definitions. If your design uses a device out of one of these libraries, you need to position the model library containing the definition of choice earlier in the list. If your system is configured as originally shipped, this means you need to add the specific library to the list *before* NOM.LIB.

Caution—Do not edit NOM.LIB. If you do, PSpice will recreate the indexes for every model library referenced in NOM.LIB. This can take some time.

Changing the library search path

For model libraries that are configured without explicit path names, PSpice first searches the directory where the current design resides, then steps down the list of directories specified in the Library Path text box on the Libraries tab of the Simulation Settings dialog box.



To change the library search path

- 1 From the Simulation menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.
- 2 Click the Libraries tab.
- 3 In the Library Path text box, position the pointer after the directory path that PSpice should search before the new path.
- 4 Type in the new path name following these rules:
 - Use a semi-colon character (;) to separate two path names.
 - Do not follow the last path name with a semi-colon.

Example: To search first C:\ORCAD\LIB, then C:\MYLIBS, for model libraries, type "C:\ORCAD\LIB"; "C:\MYLIBS" in the Library Path text box.

Creating parts for models

5

Chapter overview

This chapter provides information about creating parts for model definitions, so you can simulate the model from your design using OrCAD Capture.

For general information about creating parts, refer to the *OrCAD Capture User's Guide*.

Topics are grouped into four areas introduced later in this overview. If you want to find out quickly which tools to use to complete a given task and how to start, then:

- 1 Go to the roadmap in [Ways to create parts for models on page 5-129](#).
- 2 Find the task you want to complete.
- 3 Go to the sections referenced for that task for more information about how to proceed.

Background information These sections provide background on the things you need to know and do to prepare for creating parts:

- [What's different about parts used for simulation? on page 5-129](#)
- [Preparing your models for part creation on page 5-130](#)

Task roadmap This section helps you find the sections in this chapter that are relevant to the part creation task that you want to complete:

- [Ways to create parts for models on page 5-129](#)

How to use the tools These sections explain how to use different tools to create parts for model definitions:

- [Using the Model Editor to create parts on page 5-131](#)
- [Using the Model Editor to create parts on page 5-131](#)
- [Basing new parts on a custom set of parts on page 5-133](#)

Other useful information These sections explain how to refine part graphics and properties:

- [Editing part graphics on page 5-135](#)
- [Defining part properties needed for simulation on page 5-139](#)

What's different about parts used for simulation?

A part used for simulation has these special characteristics:

- a link to a simulation model
- a netlist translation
- modeled pins

For information on adding simulation models to a model library, see [Chapter 4, Creating and editing models](#).

Ways to create parts for models

If you want to...	Then do this...	To find out more, see this...
<ul style="list-style-type: none"> ➔ Create parts for a set of vendor or user-defined models saved in a model library. ➔ Change the graphic standard for an existing model library. 	Use the Model Editor to create parts from a model library.	Basing new parts on a custom set of parts on page 5-133
<ul style="list-style-type: none"> ➔ Automatically create one part each time you extract a new model. 	Use the Model Editor* and enable automatic creation of parts.	Using the Model Editor to create parts on page 5-131 Using the Model Editor to edit models on page 4-93 Basing new parts on a custom set of parts on page 5-133

* For a list of device types that the Model Editor supports, see [Model Editor-supported device types on page 4-95](#).

Preparing your models for part creation

If you already have model definitions and want to create parts for them, you should organize the definitions into libraries containing similar device types.

To set up a model library for part creation

- 1 If all of your models are in one file and you wish to keep them that way, rename the file to:
 - Reflect the kinds of models contained in the file.
 - Have the .LIB extension.
- 2 If each model is in its own file, and you want to concatenate them into one file, use the DOS copy command.

Example: You can append a set of files with .MOD extensions into a single .LIB file using the DOS command:

```
copy *.MOD MYLIB.LIB
```

- 3 Make sure the model names in your new library do not conflict with model names in any other model library.

Model libraries typically have a .LIB extension. However, you can use a different file extension as long as the file format conforms to the standard model library file format.

For information on managing model libraries, including the search order PSpice uses, see [Configuring model libraries on page 4-120](#).

Using the Model Editor to create parts

If you want to run the Model Editor and enable automatic creation of parts for any model that you create or change, then run the Model Editor alone. This means any models you create are not tied to the current design or to a part editing session.

Note *If you open an existing model library, the Model Editor creates parts for only the models that you change or add to it.*

To find out how to use the Model Editor to create models, see [Using the Model Editor to edit models on page 4-93](#).

To find out which device types the Model Editor supports, see [Model Editor-supported device types on page 4-95](#).

Starting the Model Editor

To start the Model Editor alone

- 1 From the Windows Start menu, point to the OrCAD Release 9 program folder, then choose PSpice Model Editor.
- 2 From the File menu, choose Open or New, and enter an existing or new model library name.
- 3 In the Models List frame, select the name of a model to display it for editing in the Spec Entry frame.

To start the Model Editor from within Capture

- 1 In the schematic page editor, select the part whose model you want to edit.
- 2 From the Edit menu, choose PSpice Model.
The Model Editor starts with the model loaded for editing.

If you have already started the Model Editor from Capture, and want to continue working on new models and parts, then:

- 1 Close the opened model library.
- 2 Open a new model library.
- 3 Load a device model or create a new one.

Setting up automatic part creation

Part creation from the Model Editor is optional. By default, automatic part creation is enabled. However, if you previously disabled part creation, you need to enable it before creating a new model and part.

Instead of using the OrCAD default part set, you can use your own set of standard parts. To find out more, see [Basing new parts on a custom set of parts on page 5-133](#).

For example, if the model library is named MYPARTS.LIB, then the Model Editor creates the part library named MYPARTS.OLB.

To automatically create parts for new models

- 1 In the Model Editor, from the Tools menu, choose Options.
- 2 In the Part Creation Setup frame, select Create Parts for Models if it is not already enabled.
- 3 In the Save Part To frame, define the name of the part library for the new part. Choose one of the following:
 - Part library path same as model library to create or open the *.OLB file that has the same filename as the open model library (*.LIB).
 - User-defined part library, and then enter a library name in the part Library Name text box.

Basing new parts on a custom set of parts

If you are using the the Model Editor to automatically generate parts for model definitions, and you want to base the new parts on a custom graphic standard (rather than the OrCAD default parts), then you can change which underlying parts either application uses by setting up your own set of parts.

Note If you use a custom part set, the Model Editor always checks the custom part library first for a part that matches the model definition. If none can be found, they use the OrCAD default part instead.

To create a custom set of parts for automatic part generation

- 1 Create a part library with the custom parts.

Be sure to name these parts by their device type as shown in Table 2; this is how the Model Editor determines which part to use for a model definition.

For more information on creating parts, refer to the *OrCAD Capture User's Guide*.

Table 2 Part names for custom part generation.

For this device type...	Use this part name...	For this device type...	Use this part name...
Bipolar transistor: LPNP	LPNP	MOSFET: N-channel	NMOS
Bipolar transistor: NPN	NPN	MOSFET: P-channel	PMOS
Bipolar transistor: PNP	PNP	OPAMP: 5-pin	OPAMP5
Capacitor*	CAP	OPAMP: 7-pin	OPAMP7
Diode	DIODE	Resistor*	RES
GaAsFET*	GASFET	Switch: voltage-controlled*	VSWITCH
IGBT: N-channel	NIGBT	Transmission line*	TRN
Inductor*	IND	Voltage comparator	VCOMP
JFET: N-channel	NJF	Voltage comparator: 6 pin	VCOMP6
JFET: P-channel	PJF	Voltage reference	VREF
Magnetic core	CORE	Voltage regulator	VREG

* Does not apply to the Model Editor.

- 2 For each custom part, set its MODEL property to `M where ` is a back-single quote or grave character.

This tells the Model Editor to substitute the correct model name.

To base new parts on custom parts using the Model Editor

- 1 In the Model Editor, from the Options menu, choose Part Creation Setup, and enable automatic part creation as described in [To automatically create parts for new models on page 5-132](#).
- 2 In the Base Parts On frame, enter the name of the existing part library (*.OLB) that contains your custom parts.
- 3 Click OK.

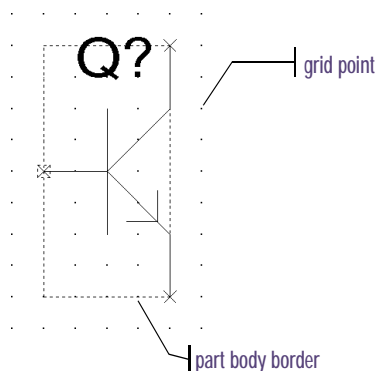
Editing part graphics

If you created parts using the Model Editor, and you want to make further changes, the following sections explain a few important things to remember when you edit the parts.

When changing part graphics, check to see that all pins are on the grid.

How Capture places parts

When placing parts on the schematic page, the schematic page editor uses the grid as a point of reference for different editing activities. The part's pin ends are positioned on the grid points.



To edit a part in a library

- 1 From Capture's File menu, point to Open, then choose Library.
- 2 Select the library that has the part you want to edit.
The library opens and displays all its parts.
- 3 Double-click the part you want to edit.
The part appears in the part editor.
- 4 Edit the part.
You can resize it, add or delete graphics, and add or delete pins.

For more information about specific part editing tasks, refer to the *OrCAD Capture User's Guide*.

- 5 After you have finished editing the part, from the File menu, choose Save to save the part to its library.

Defining grid spacing

Grid spacing for graphics

The grid, denoted by evenly spaced grid points, regulates the sizing and positioning of graphic objects and the positioning of pins. The default grid spacing with snap-to-grid enabled is 0.10", and the grid spacing is 0.01".

You can turn off the grid spacing when you need to draw graphics in a tighter space.

To edit the part graphics

- 1 In Capture's part editor, display the part you want to edit.
- 2 Select the line, arc, circle, or other graphic object you want to change, and do any of the following:
 - To stretch or shrink the graphic object, click and drag one of the size handles.
 - To move the entire part graphic, click and drag the edge of the part.

The part body border automatically changes to fit the size of the part graphic.

- 3 After you have finished editing the part, from the File menu, choose Save to save the part to its library.

Note Pin changes that alter the part template can occur if you either:

- change pin names

or

- delete pins

In these cases you must adjust the value of the part's PSPICETEMPLATE property to reflect these changes. To find out how, see

[Pin callout in subcircuit templates on page 5-144.](#)

Grid spacing for pins

The part editor always places pins on the grid, even when the snap-to-grid option is turned off. The size of the part is relative to the pin-to-pin spacing for that part. That means that pins placed one grid space apart in the part editor are displayed as one grid space apart in the schematic page editor.

Pins *must* be placed on the grid at integer multiples of the grid spacing. Because the default grid spacing for the Schematic Page Grid is set at 0.10", OrCAD recommends setting pin spacing in the Part and Symbol Grid at 0.10" intervals from the origin of the part and at least 0.10" from any adjacent pins.

The part editor considers pins that are not placed at integer multiples of the grid spacing from the origin as *off-grid*, and a warning appears when you try to save the part.

Here are two guidelines:

- Make sure Pointer Snap to Grid is enabled when editing part pins and editing schematic pages so you can easily make connections.
- Make sure the Part and Symbol Grid spacing *matches* the Schematic Page Grid spacing.

For more information about grid spacing and pin placement, refer to the *OrCAD Capture User's Guide*.

Attaching models to parts

If you create parts and want to simulate them, you need to attach model implementations to them. If you created your parts using any of the methods discussed in this chapter, then your part will have a model implementation already attached to it.

MODEL

The Implementation property defines the name of the model that PSpice must use for simulation. When attaching this implementation, this rule applies:

- The Implementation name should match the name of the .MODEL or .SUBCKT definition of the simulation model as it appears in the model library (*.LIB).

Example: If your design includes a 2N2222 bipolar transistor with a .MODEL name of Q2N2222, then the Implementation name for that part should be Q2N2222.

Note *Make sure that the model library containing the definition for the attached model is configured in the list of libraries for your project. See [Configuring model libraries on page 4-120](#) for more information.*

For more information on model editing in general, see [Chapter 4, Creating and editing models](#). For specific information on changing model references, see [Changing the model reference to an existing model definition on page 4-117](#).

You do not need to enter an Implementation Path because PSpice searches for the model in the list of model libraries you configure for this project.

To attach a model implementation

- 1 In the schematic page editor, double-click a part to display the Parts spreadsheet of the Property Editor.
- 2 From the Implementation list, select PSpice Model.
- 3 In the Implementation column, type the name of the model to attach to the part.
- 4 Click Apply to update the design, then close the Parts spreadsheet.

Defining part properties needed for simulation

If you created your parts using any of the methods discussed in this chapter, then your part will have these properties already defined for it:

- PSpice PSPICETEMPLATE for simulation
- PART and REFDES for identification

For example, if you create a part that has electrical behavior described by the subcircuit definition that starts with:

```
.SUBCKT 7400 A B Y
+ optional: DPWR=$G_DPWR DGND=$G_DGND
+ params: MNTYMXDLY=0 IO_LEVEL=0
```

then the appropriate part properties are:

```
IMPLEMENTATION = 7400
MNTYMXDLY = 0
IO_LEVEL = 0
PSPICETEMPLATE = X^@REFDES %A %B %Y %PWR
%GND
@MODEL PARAMS:IO_LEVEL=@IO_LEVEL
MNTYMXDLY=@MNTYMXDLY
```

Note For clarity, the PSPICETEMPLATE property value is shown here in multiple lines; in a part definition, it is specified in one line (no line breaks).

Table 3

To find out more about this property...	See this...
PSPICETEMPLATE	page 5-140

Here are the things to check when editing part properties:

- ✓ Does the PSPICETEMPLATE specify the correct number of pins/nodes?
- ✓ Are the pins/nodes in the PSPICETEMPLATE specified in the proper order?
- ✓ Do the pin/node names in the PSPICETEMPLATE match the pin names on the part?

To edit a property needed for simulation:

- 1 In the schematic page editor, select the part to edit.
- 2 From the Edit menu, choose Properties to display the Parts spreadsheet of the Property Editor.
- 3 Click in the cell of the column you want to change (for example, PSPICETEMPLATE), or click the New button to add a property (and type the property name in the Name text box).
- 4 If needed, type a value in the Value text box.
- 5 Click Apply to update the design, then close the spreadsheet.

Caution—Creating parts not intended for simulation

Some part libraries contain parts designed only for board layout; PSpice cannot simulate these parts. This means they do not have PSPICETEMPLATE properties or that the PSPICETEMPLATE property value is blank.

PSPICETEMPLATE

The PSPICETEMPLATE property defines the PSpice syntax for the part's netlist entry. When creating a netlist, Capture substitutes actual values from the circuit into the appropriate places in the PSPICETEMPLATE syntax, then saves the translated statement to the netlist file.

Any part that you want to simulate must have a defined PSPICETEMPLATE property. These rules apply:

- The pin names specified in the PSPICETEMPLATE property must match the pin names on the part.
- The number and order of the pins listed in the PSPICETEMPLATE property must match those for the associated .MODEL or .SUBCKT definition referenced for simulation.
- The first character in a PSPICETEMPLATE must be a PSpice device letter appropriate for the part (such as Q for a bipolar transistor).

PSPICETEMPLATE syntax

The PSPICETEMPLATE contains:

- *regular characters* that the schematic page editor interprets verbatim
- *property names and control characters* that the schematic page editor translates

Regular characters in templates

Regular characters include the following:

- alphanumerics
- any keyboard part *except* the special syntactical parts used with properties (@ & ? ~ #).
- white space

An *identifier* is a collection of regular characters of the form:

alphabetic character [any other regular character].*

Property names in templates

Property names are preceded by a special character as follows:

[@ | ? | ~ | # | &]<identifier>

The schematic page editor processes the property according to the special character as shown in the following table.

Table 4

This syntax...*	Is replaced with this...
@<id>	Value of <id>. Error if no <id> attribute or if no value assigned.
&<id>	Value of <id> if <id> is defined.
?<id>s...s	Text between s...s separators if <id> is defined.
?<id>s...ss...s	Text between the first s...s separators if <id> is defined, else the second s...s clause.
~<id>s...s	Text between s...s separators if <id> is undefined.
~<id> s...ss...s	Text between the first s...s separators if <id> is undefined, else the second s...s clause.
#<id>s...s	Text between s...s separators if <id> is defined, but delete rest of template if <id> is undefined.

* s is a separator character

Separator characters include commas (,), periods (.), semi-colons (;), forward slashes (/), and vertical bars (|). You must always use the same character to specify an opening-closing pair of separators.

Note You can use different separator characters to nest conditional property clauses.

Example: The template fragment
 ?G|G=@G||G=1000| uses the vertical bar as the separator between the if-then-else parts of this conditional clause. If G has a value, then this fragment translates to G=<G property value>. Otherwise, this fragment translates to G=1000.

Caution—Recommended scheme for netlist templates

Templates for devices in the part library start with a PSpice device letter, followed by the hierarchical path, and then the reference designator (REFDES) property. OrCAD recommends that you adopt this scheme when defining your own netlist templates.

Example: R^@REFDES ... for a resistor

The ^ character in templates

The schematic page editor replaces the ^ character with the complete hierarchical path to the device being netlisted.

The \n character sequence in templates

The part editor replaces the character sequence \n with a new line. Using \n, you can specify a multi-line netlist entry from a one-line template.

The % character and pin names in templates

Pin names are denoted as follows:

%<pin name>

where *pin name* is one or more regular characters.

The schematic page editor replaces the %<pin name> clause in the template with the name of the net connected to that pin.

The end of the pin name is marked with a separator (see [Property names in templates on page 5-141](#)). To avoid name conflicts in PSpice, the schematic page editor translates the following characters contained in pin names.

Table 5

This pin name character...	Is replaced with this...
<	l (L)
>	g
=	e
\XXX\	XXXbar

Note To include a literal % character in the netlist, type %% in the template.

PSPICETEMPLATE examples

Simple resistor (R) template

The R part has:

- two pins: 1 and 2
- two required properties: REFDES and VALUE

Template

```
R^@REFDES %1 %2 @VALUE
```

Sample translation

```
R_R23 abc def 1k
```

where REFDES equals R23, VALUE equals 1k, and R is connected to nets abc and def.

Voltage source with optional AC and DC specifications (VAC) template

The VAC part has:

- two properties: AC and DC
- two pins: + and -

Template

```
V^@REFDES %+ %- ?DC | DC=@DC | ?AC | AC=@AC |
```

Sample translation

```
V_V6 vp vm DC=5v
```

where REFDES equals V6, VSRC is connected to nodes vp and vm, DC is set to 5v, and AC is undefined.

Sample translation

```
V_V6 vp vm DC=5v AC=1v
```

where, in addition to the settings for the previous translation, AC is set to 1v.

Parameterized subcircuit call (X) template

Suppose you have a subcircuit Z that has:

- two pins: a and b
- a subcircuit parameter: G, where G defaults to 1000 when no value is supplied

To allow the parameter to be changed on the schematic page, treat G as an property in the template.

Note For clarity, the PSPICETEMPLATE property value is shown here in multiple lines; in a part definition, it is specified in one line (no line breaks).

Template

```
X^@REFDES %a %b Z PARAMS: ?G|G=@G|
~G|G=1000|
```

Equivalent template (using the if...else form)

```
X^@REFDES %a %b Z PARAMS: ?G|G=@G| |G=1000|
```

Sample translation

```
X_U33 101 102 Z PARAMS: G=1024
```

where REFDES equals U33, G is set to 1024, and the subcircuit connects to nets 101 and 102.

Sample translation

```
X_U33 101 102 Z PARAMS: G=1000
```

where the settings of the previous translation apply except that G is undefined.

Pin callout in subcircuit templates

To find out how to define subcircuits, refer to the .SUBCKT command in the online *OrCAD PSpice A/D Reference Manual*.

The number and sequence of pins named in a template for a subcircuit must agree with the definition of the subcircuit itself—that is, the node names listed in the .SUBCKT statement, which heads the definition of a subcircuit. These are the pinouts of the subcircuit.

Example: Consider the following first line of a (hypothetical) subcircuit definition:

```
.SUBCKT SAMPLE 10 3 27 2
```

The four numbers following the name SAMPLE—10, 3, 27, and 2—are the node names for this subcircuit's pinouts.

Now suppose that the part definition shows four pins:

IN+ OUT+ IN- OU
T-

The number of pins on the part equals the number of nodes in the subcircuit definition.

If the correspondence between pin names and nodes is as follows:

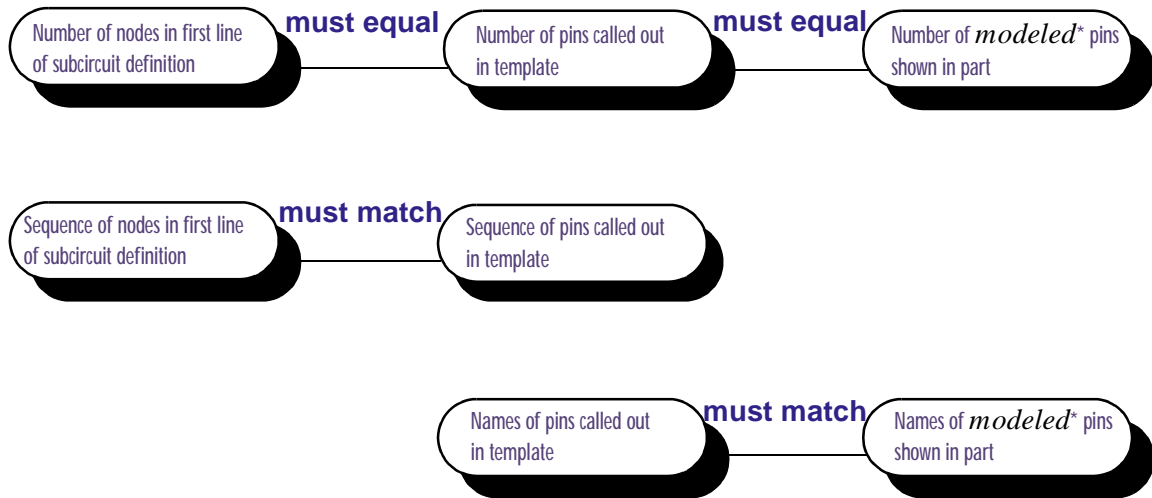
Table 6

This node name...	Corresponds to this pin name...
10	IN+
3	IN-
27	OUT+
2	OUT-

then the template looks like this:

```
X^@REFDES %IN+ %IN- %OUT+ %OUT- @MODEL
```

The *rules of agreement* are outlined in Figure 34.



* Unmodeled pins may appear on a part (like the two voltage offset pins on a 741 opamp part). These pins are not netlisted and do not appear on the template.

Figure 34 *Rules for pin callout in subcircuit templates.*

Analog behavioral modeling

6

Chapter overview

This chapter describes how to use the Analog Behavioral Modeling (ABM) feature of PSpice. This chapter includes the following sections:

- [Overview of analog behavioral modeling on page 6-148](#)
- [The ABM.OLB part library file on page 6-149](#)
- [Placing and specifying ABM parts on page 6-150](#)
- [ABM part templates on page 6-152](#)
- [Control system parts on page 6-153](#)
- [PSpice-equivalent parts on page 6-174](#)
- [Cautions and recommendations for simulation and analysis on page 6-186](#)
- [Basic controlled sources on page 6-192](#)

Overview of analog behavioral modeling

You can use the Analog Behavioral Modeling (ABM) feature of PSpice to make flexible descriptions of electronic components in terms of a transfer function or lookup table. In other words, a mathematical relationship is used to model a circuit segment, so you do not need to design the segment component by component.

The part library contains several ABM parts that are classified as either control system parts or as PSpice-equivalent parts. See [Basic controlled sources on page 6-192](#) for an introduction to these parts, how to use them, and the difference between parts with general-purpose application and parts with special-purpose application.

Control system parts are defined with the reference voltage preset to ground so that each controlling input and output are represented by a single pin in the part. These are described in [Control system parts on page 6-153](#).

PSpice-equivalent parts reflect the structure of the PSpice E and G device types, which respond to a differential input and have double-ended output. These are described in [PSpice-equivalent parts on page 6-174](#).

You can also use the Device Equations option (described in the online *OrCAD PSpice A/D Reference Manual*) for modeling of this type, but OrCAD recommends using the ABM feature wherever possible. With Device Equations, the PSpice source code is actually modified. While this is more flexible and produces faster results, it is also much more difficult to use and to troubleshoot. Also, any changes you make using Device Equations must be made to all new PSpice updates you install.

Device models made with ABM can be used for most cases, are much easier to create, and are compatible with PSpice updates.

The ABM.OLB part library file

The part library ABM.OLB contains the ABM components. This library contains two sections.

The first section has parts that you can quickly connect to form control system types of circuits. These components have names like SUM, GAIN, LAPLACE, and HIPASS.

The second section contains parts that are useful for more traditional controlled source forms of schematic parts. These PSpice-equivalent parts have names like EVALUE and GFREQ and are based on extensions to traditional PSpice E and G device types.

Implement ABM components by using PSpice primitives; there is no corresponding `abm.lib` model library. A few components generate multi-line netlist entries, but most are implemented as single PSpice E or G device declarations. See [ABM part templates on page 6-152](#) for a description of PSPICETEMPLATE properties and their role in generating netlist declarations. See [Implementation of PSpice -equivalent parts on page 6-175](#) for more information about PSpice E and G syntax.

Placing and specifying ABM parts

Place and connect ABM parts the same way you place other parts. After you place an ABM part, you can edit the instance properties to customize the operational behavior of the part. This is equivalent to defining an ABM expression describing how inputs are transformed into outputs. The following sections describe the rules for specifying ABM expressions.

Net names and device names in ABM expressions

In ABM expressions, refer to signals by name. This is also considerably more convenient than having to connect a wire from a pin on an ABM component to a point carrying the voltage of interest.

The name of an interface port does not extend to any connected nets. To refer to a signal originating at an interface port, connect the port to an offpage connector of the desired name.

If you used an expression such as `V(2)`, then the referenced net (2 in this case) is interpreted as the name of a local or global net. A local net is a labeled wire or bus segment in a hierarchical schematic, or a labeled offpage connector. A global net is a labeled wire or bus segment at the top level, or a global connector.

OrCAD Capture recognizes these constructs in ABM expressions:

```
V (<net name>)
V (<net name>, <net name>)
I (<vdevice>)
```

When one of these is recognized, Capture searches for `<net name>` or `<vdevice>` in the net name space or the device name space, respectively. Names are searched for first at the hierarchical level of the part being netlisted. If not found there, then the set of global names is searched. If the fragment is not found, then a warning is issued but Capture still outputs the resulting netlist. When a match is

found, the original fragment is replaced by the fully qualified name of the net or device.

For example, suppose we have a hierarchical part U1. Inside the schematic representing U1 we have an ABM expression including the term V(Reference). If “Reference” is the name of a local net, then the fragment written to the netlist will be translated to V(U1_Reference). If “Reference” is the name of a global net, the corresponding netlist fragment will be V(Reference).

Names of voltage sources are treated similarly. For example, an expression including the term I(Vsense) will be output as I(V_U1_Vsense) if the voltage source exists locally, and as I(V_Vsense) if the voltage source exists at the top level.

Forcing the use of a global definition

If a net name exists both at the local hierarchical level and at the top level, the search mechanism used by Capture will find the local definition. You can override this, and force Capture to use the global definition, by prefixing the name with a single quote (') character.

For example, suppose there is a net called Reference both inside hierarchical part U1 and at the top level. Then, the ABM fragment V(Reference) will result in V(U1_Reference) in the netlist, while the fragment V('Reference) will produce V(Reference).

ABM part templates

For most ABM parts, a single PSpice “E” or “G” device declaration is output to the netlist per part instance. The PSPICETEMPLATE property in these cases is straightforward. For example the LOG part defines an expression variant of the E device with its output being the natural logarithm of the voltage between the input pin and ground:

```
E^@REFDES %out 0 VALUE { LOG(V(%in)) }
```

The fragment E^@REFDES is standard. The “E” specifies a PSpice controlled voltage source (E device); %in and %out are the input and output pins, respectively; VALUE is the keyword specifying the type of ABM device; and the expression inside the curly braces defines the logarithm of the input voltage.

Several ABM parts produce more than one primitive PSpice device per part instance. In this case, the PSPICETEMPLATE property may be quite complicated. An example is the DIFFER (differentiator) part. This is implemented as a capacitor in series with a current sensor together with an E device which outputs a voltage proportional to the current through the capacitor.

The template has several unusual features: it gives rise to three primitives in the PSpice netlist, and it creates a local node for the connection of the capacitor and its current-sensing V device.

For clarity, the template is shown on three lines although the actual template is a single line.

```
C^@REFDES %in $$U^@REFDES 1\n
V^@REFDES $$U^@REFDES 0 0v\n
E^@REFDES %out 0 VALUE {@GAIN * I(V^@REFDES)}
```

The fragments C^@REFDES, V^@REFDES, and E^@REFDES create a uniquely named capacitor, current sensing V device, and E device, respectively. The fragment \$\$U^@REFDES creates a name suitable for use as a local node. The E device generates an output proportional to the current through the local V device.

Control system parts

Control system parts have single-pin inputs and outputs. The reference for input and output voltages is analog ground (0). An enhancement to PSpice means these components can be connected together with no need for dummy load or input resistors.

Table 7 lists the control system parts, grouped by function. Also listed are characteristic properties that may be set. In the sections that follow, each part and its properties are described in more detail.

Table 7 *Control system parts*

Category	Part	Description	Properties
Basic components	CONST	constant	VALUE
	SUM	adder	
	MULT	multiplier	
	GAIN	gain block	GAIN
	DIFF	subtractor	
Limiters	LIMIT	hard limiter	LO, HI
	GLIMIT	limiter with gain	LO, HI, GAIN
	SOFTLIM	soft (tanh) limiter	LO, HI, GAIN
Chebyshev filters	LOPASS	lowpass filter	FP, FS, RIPPLE, STOP
	HIPASS	highpass filter	FP, FS, RIPPLE, STOP
	BANDPASS	bandpass filter	F0, F1, F2, F3, RIPPLE, STOP
	BANDREJ	band reject (notch) filter	F0, F1, F2, F3, RIPPLE, STOP
Integrator and differentiator	INTEG	integrator	GAIN, IC
	DIFFER	differentiator	GAIN
Table look-ups	TABLE	lookup table	ROW1...ROW5
	FTABLE	frequency lookup table	ROW1...ROW5

Table 7 Control system parts (continued)

Category	Part	Description	Properties
Laplace transform	LAPLACE	Laplace expression	NUM, DENOM
Math functions (where 'x' is the input)	ABS	$ x $	
	SQRT	$x^{1/2}$	
	PWR	$ x ^{\text{EXP}}$	EXP
	PWRS	x^{EXP}	EXP
	LOG	$\ln(x)$	
	LOG10	$\log(x)$	
	EXP	e^x	
	SIN	$\sin(x)$	
	COS	$\cos(x)$	
	TAN	$\tan(x)$	
	ATAN	$\tan^{-1}(x)$	
	ARCTAN	$\tan^{-1}(x)$	
Expression functions	ABM	no inputs, V out	EXP1...EXP4
	ABM1	1 input, V out	EXP1...EXP4
	ABM2	2 inputs, V out	EXP1...EXP4
	ABM3	3 inputs, V out	EXP1...EXP4
	ABM/I	no input, I out	EXP1...EXP4
	ABM1/I	1 input, I out	EXP1...EXP4
	ABM2/I	2 inputs, I out	EXP1...EXP4
	ABM3/I	3 inputs, I out	EXP1...EXP4

Basic components

The basic components provide fundamental functions and in many cases, do not require specifying property values. These parts are described below.

CONST

VALUE constant value

The CONST part outputs the voltage specified by the VALUE property. This part provides no inputs and one output.

SUM

The SUM part evaluates the voltages of the two input sources, adds the two inputs together, then outputs the sum. This part provides two inputs and one output.

MULT

The MULT part evaluates the voltages of the two input sources, multiplies the two together, then outputs the product. This part provides two inputs and one output.

GAIN

GAIN constant gain value

The GAIN part multiplies the input by the constant specified by the GAIN property, then outputs the result. This part provides one input and one output.

DIFF

The DIFF part evaluates the voltage difference between two inputs, then outputs the result. This part provides two inputs and one output.

Limiters

The Limiters can be used to restrict an output to values between a set of specified ranges. These parts are described below.

LIMIT

HI	upper limit value
LO	lower limit value

The LIMIT part constrains the output voltage to a value between an upper limit (set with the HI property) and a lower limit (set with the LO property). This part takes one input and provides one output.

GLIMIT

HI	upper limit value
LO	lower limit value
GAIN	constant gain value

The GLIMIT part functions as a one-line opamp. The gain is applied to the input voltage, then the output is constrained to the limits set by the LO and HI properties. This part takes one input and provides one output.

SOFTLIMIT

HI	upper limit value
LO	lower limit value
GAIN	constant gain value
A, B, V, TANH	internal variables used to define the limiting function

The SOFTLIMIT part provides a limiting function much like the LIMIT device, except that it uses a continuous curve limiting function, rather than a discontinuous limiting function. This part takes one input and provides one output.

Chebyshev filters

The Chebyshev filters allow filtering of the signal based on a set of frequency characteristics. The output of a Chebyshev filter depends upon the analysis being performed.

Note *PSpice computes the impulse response of each Chebyshev filter used in a transient analysis during circuit read-in. This may require considerable computing time. A message is displayed on your screen indicating that the computation is in progress.*

For DC and bias point, the output is simply the DC response of the filter. For AC analysis, the output for each frequency is the filter response at that frequency. For transient analysis, the output is then the convolution of the past values of the input with the impulse response of the filter. These rules follow the standard method of using Fourier transforms.

Note *To obtain a listing of the filter Laplace coefficients for each stage, choose Setup from the Analysis menu, click on Options, and enable LIST in the Options dialog box.*

Each of the Chebyshev filter parts is described in the following pages.

LOPASS

FS	stop band frequency
FP	pass band frequency
RIPPLE	pass band ripple in dB
STOP	stop band attenuation in dB

The LOPASS part is characterized by two cutoff frequencies that delineate the boundaries of the filter pass band and stop band. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The LOPASS part provides one input and one output.

Figure 35 shows an example of a LOPASS filter device. The filter provides a pass band cutoff of 800 Hz and a stop

OrCAD Capture recommends looking at one or more of the references cited in [Frequency-domain device models on page 6-181](#), as well as some of the following references on analog filter design:

- 1 Ghavsi, M.S. & Laker, K.R., *Modern Filter Design*, Prentice-Hall, 1981.
- 2 Gregorian, R. & Temes, G., *Analog MOS Integrated Circuits*, Wiley-Interscience, 1986.
- 3 Johnson, David E., *Introduction to Filter Theory*, Prentice-Hall, 1976.
- 4 Lindquist, Claude S., *Active Network Design with Signal Filtering Applications*, Steward & Sons, 1977.
- 5 Stephenson, F.W. (ed), *RC Active Filter Design Handbook*, Wiley, 1985.
- 6 Van Valkenburg, M.E., *Analog Filter Design*, Holt, Rinehart & Winston, 1982.
- 7 Williams, A.B., *Electronic Filter Design Handbook*, McGraw-Hill, 1981.

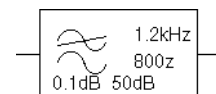


Figure 35 LOPASS filter example.

band cutoff of 1.2 kHz. The pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. Assuming that the input to the filter is the voltage at net 10 and output is a voltage between nets 5 and 0, this will produce a PSpice netlist declaration like this:

```
ELOWPASS 5 0 CHEBYSHEV {V(10)} = LP 800 1.2K .1dB 50dB
```

HIPASS

FS	stop band frequency
FP	pass band frequency
RIPPLE	pass band ripple in dB
STOP	stop band attenuation in dB

The HIPASS part is characterized by two cutoff frequencies that delineate the boundaries of the filter pass band and stop band. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The HIPASS part provides one input and one output.

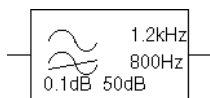


Figure 36 HIPASS filter part example.

Figure 36 shows an example of a HIPASS filter device. This is a high pass filter with the pass band above 1.2 kHz and the stop band below 800 Hz. Again, the pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. This will produce a PSpice netlist declaration like this:

```
EHIGHPASS 5 0 CHEBYSHEV {V(10)} = HP 1.2K 800 .1dB 50dB
```

BANDPASS

RIPPLE	pass band ripple in dB
STOP	stop band attenuation in dB
F0, F1, F2, F3	cutoff frequencies

The BANDPASS part is characterized by four cutoff frequencies. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop

band, respectively. The BANDPASS part provides one input and one output.

Figure 37 shows an example of a BANDPASS filter device. This is a band pass filter with the pass band between 1.2 kHz and 2 kHz, and stop bands below 800 Hz and above 3 kHz. The pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. This will produce a PSpice netlist declaration like this:

```
EBANDPASS 5 0 CHEBYSHEV
+ {V(10)} = BP 800 1.2K 2K 3K .1dB 50dB
```

BANDREJ

RIPPLE is the pass band ripple in dB
 STOP is the stop band attenuation in dB
 F0, F1, are the cutoff frequencies
 F2, F3

The BANDREJ part is characterized by four cutoff frequencies. The attenuation values, RIPPLE and STOP, define the maximum allowable attenuation in the pass band, and the minimum required attenuation in the stop band, respectively. The BANDREJ part provides one input and one output.

Figure 38 shows an example of a BANDREJ filter device. This is a band reject (or “notch”) filter with the stop band between 1.2 kHz and 2 kHz, and pass bands below 800 Hz and above 3 kHz. The pass band ripple is 0.1 dB and the minimum stop band attenuation is 50 dB. This will produce a PSpice netlist declaration like this:

```
ENOTCH 5 0 CHEBYSHEV {V(10)} = BR 1.2K 800 3K 2K .1dB 50dB
```

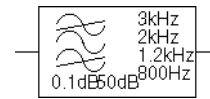


Figure 37 BANDPASS filter part example.

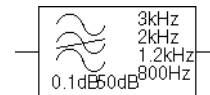


Figure 38 BANDREJ filter part example.

Integrator and differentiator

The integrator and differentiator parts are described below.

INTEG

IC	initial condition of the integrator output
GAIN	gain value

The INTEG part implements a simple integrator. A current source/capacitor implementation is used to provide support for setting the initial condition.

DIFFER

GAIN	gain value
------	------------

The DIFFER part implements a simple differentiator. A voltage source/capacitor implementation is used. The DIFFER part provides one input and one output.

Table look-up parts

TABLE and FTABLE parts provide a lookup table that is used to correlate an input and an output based on a set of data points. These parts are described below and on the following pages.

TABLE

If more than five values are required, the part can be customized through the part editor. Insert additional row variables into the template using the same form as the first five, and add ROW n properties as needed to the list of properties.

ROW n	is an (input, output) pair; by default, up to five triplets are allowed where $n=1, 2, 3, 4,$ or 5
---------	--

The TABLE part allows the response to be defined by a table of one to five values. Each row contains an input and a corresponding output value. Linear interpolation is performed between entries.

For values outside the table's range, the device's output is a constant with a value equal to the entry with the smallest (or largest) input. This characteristic can be used to

impose an upper and lower limit on the output. The TABLE part provides one input and one output.

FTABLE

ROW n	either an (input frequency, magnitude, phase) triplet, or an (input frequency, real part, imaginary part) triplet describing a complex value; by default, up to five triplets are allowed where $n=1, 2, 3, 4$, or 5
DELAY	group delay increment; defaults to 0 if left blank
R_I	table type; if left blank, the frequency table is interpreted in the (input frequency, magnitude, phase) format; if defined with any value (such as YES), the table is interpreted in the (input frequency, real part, imaginary part) format
MAGUNITS	units for magnitude where the value can be DB (decibels) or MAG (raw magnitude); defaults to DB if left blank
PHASEUNITS	units for phase where the value can be DEG (degrees) or RAD (radians); defaults to DEG if left blank

If more than five values are required, the part can be customized through the part editor. Insert additional row variables into the template using the same form as the first five, and add ROW n properties as needed to the list of properties.

The FTABLE part is described by a table of frequency responses in either the magnitude/phase domain (R_I=) or complex number domain (R_I=YES). The entire table is read in and converted to magnitude in dB and phase in degrees.

Interpolation is performed between entries. Magnitude is interpolated logarithmically; phase is interpolated linearly. For frequencies outside the table's range, 0 (zero) magnitude is used. This characteristic can be used to impose an upper and lower limit on the output.

The DELAY property increases the group delay of the frequency table by the specified amount. The delay term is particularly useful when a frequency table device generates a non-causality warning message during a transient analysis. The warning message issues a delay

value that can be assigned to the part’s DELAY property for subsequent runs, without otherwise altering the table.

The output of the part depends on the analysis being done. For DC and bias point, the output is the zero frequency magnitude times the input voltage. For AC analysis, the input voltage is linearized around the bias point (similar to EVALUE and GVALUE parts, [Modeling mathematical or instantaneous relationships on page 6-176](#)). The output for each frequency is then the input times the gain, times the value of the table at that frequency.

For transient analysis, the voltage is evaluated at each time point. The output is then the convolution of the past values with the impulse response of the frequency response. These rules follow the standard method of using Fourier transforms. We recommend looking at one or more of the references cited in [Frequency-domain device models on page 6-181](#) for more information.

Note *The table’s frequencies must be in order from lowest to highest. The TABLE part provides one input and one output.*

Example

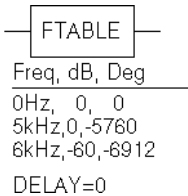


Figure 39 FTABLE part example.

A device, ELOFILT, is used as a frequency filter. The input to the frequency response is the voltage at net 10. The output is a voltage across nets 5 and 0. The table describes a low pass filter with a response of 1 (0 dB) for frequencies below 5 kilohertz and a response of 0.001 (-60 dB) for frequencies above 6 kilohertz. The phase lags linearly with frequency. This is the same as a constant time delay. The delay is necessary so that the impulse response is causal. That is, so that the impulse response does not have any significant components before time zero. The FTABLE part in Figure 39 could be used.

This part is characterized by the following properties:

```

ROW1 = 0Hz      0      0
ROW2 = 5kHz     0     -5760
ROW3 = 6kHz     -60    -6912
DELAY =
R_I =
MAGUNITS =
PHASEUNITS =

```

Since `R_I`, `MAGUNITS`, and `PHASEUNITS` are undefined, each table entry is interpreted as containing frequency, magnitude value in dB, and phase values in degrees. Delay defaults to 0.

This produces a PSpice netlist declaration like this:

```

ELOFILT 5 0 FREQ {V(10)} = (0,0,0) (5kHz,0,-5760)
+ (6kHz,-60,-6912)

```

Since constant group delay is calculated from the values for a given table entry as:

$$\text{group delay} = \text{phase} / 360 / \text{frequency}$$

An equivalent `FTABLE` instance could be defined using the `DELAY` property. For this example, the group delay is 3.2 msec ($6912 / 360 / 6k = 5760 / 360 / 6k = 3.2m$).

Equivalent property assignments are:

```

ROW1 = 0Hz      0      0
ROW2 = 5kHz     0      0
ROW3 = 6kHz     -60    0
DELAY = 3.2ms
R_I =
MAGUNITS =
PHASEUNITS =

```

This produces a PSpice netlist declaration like this:

```

ELOFILT 5 0 FREQ {V(10)} = (0,0,0) (5kHz,0,0) (6kHz,-60,0)
+ DELAY=3.2ms

```

Laplace transform part

The LAPLACE part specifies a Laplace transform which is used to determine an output for each input value.

LAPLACE

NUM	numerator of the Laplace expression
DENOM	denominator of the Laplace expression

The LAPLACE part uses a Laplace transform description. The input to the transform is a voltage. The numerator and denominator of the Laplace transform function are specified as properties for the part.

Note *Voltages, currents, and TIME may not appear in a Laplace transform specification.*

The output of the part depends on the type of analysis being done. For DC and bias point, the output is the zero frequency gain times the value of the input. The zero frequency gain is the value of the Laplace transform with $s=0$. For AC analysis, the output is then the input times the gain times the value of the Laplace transform. The value of the Laplace transform at a frequency is calculated by substituting $j\omega$ for s , where ω is 2π -frequency. For transient analysis, the output is the convolution of the input waveform with the impulse response of the transform. These rules follow the standard method of using Laplace transforms.

Example one

The input to the Laplace transform is the voltage at net 10. The output is a voltage and is applied between nets 5 and 0. For DC, the output is simply equal to the input, since the gain at $s = 0$ is 1. The transform, $1/(1+.001\cdot s)$, describes a simple, lossy integrator with a time constant of 1 millisecond. This can be implemented with an RC pair that has a time constant of 1 millisecond.

For AC analysis, the gain is found by substituting $j\omega$ for s . This gives a flat response out to a corner frequency of $1000/(2\pi) = 159$ hertz and a roll-off of 6 dB per octave after

159 Hz. There is also a phase shift centered around 159 Hz. In other words, the gain has both a real and an imaginary component. For transient analysis, the output is the convolution of the input waveform with the impulse response of $1/(1+.001 \cdot s)$. The impulse response is a decaying exponential with a time constant of 1 millisecond. This means that the output is the “lossy integral” of the input, where the loss has a time constant of 1 millisecond. The LAPLACE part shown in Figure 40 could be used for this purpose.

The transfer function is the Laplace transform $(1/[1+.001 \cdot s])$. This LAPLACE part is characterized by the following properties:

$$\begin{aligned}\text{NUM} &= 1 \\ \text{DENOM} &= 1 + .001 \cdot s\end{aligned}$$

The gain and phase characteristics are shown in Figure 41.

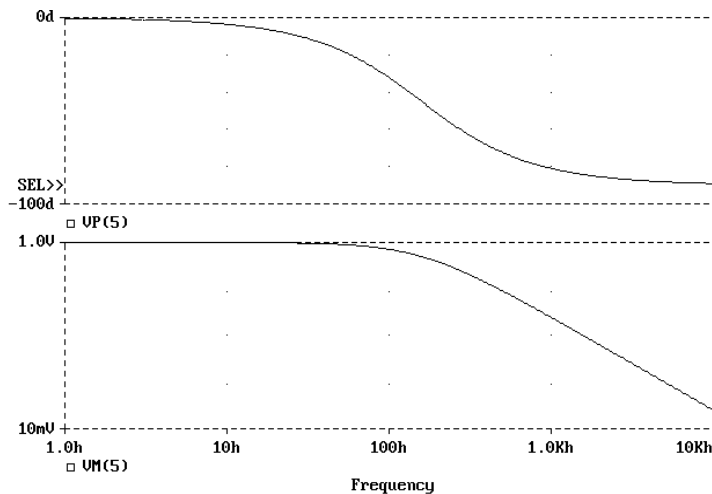


Figure 41 Viewing gain and phase characteristics of a lossy integrator.

This produces a PSpice netlist declaration like this:

```
ERC 5 0 LAPLACE {V(10)} = {1/(1+.001*s)}
```

Example two

The input is V(10). The output is a current applied between nets 5 and 0. The Laplace transform describes a

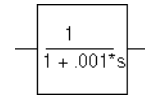


Figure 40 LAPLACE part example one.

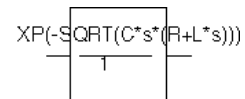


Figure 42 LAPLACE part example two.

lossy transmission line. R, L, and C are the resistance, inductance, and capacitance of the line per unit length.

If R is small, the characteristic impedance of such a line is $Z = ((R + j\omega L)/(j\omega C))^{1/2}$, the delay per unit length is $(L/C)^{1/2}$, and the loss in dB per unit length is $23 \cdot R/Z$. This could be represented by the device in Figure 42.

The parameters R, L, and C can be defined in a .PARAM statement contained in a model file. (Refer to the online *OrCAD PSpice A/D Reference Manual* for more information about using .PARAM statements.) More useful, however, is for R, L, and C to be arguments passed into a subcircuit. This part has the following characteristics:

```
NUM = EXP(-SQRT(C*s*(R+L*s)))  
DENOM = 1
```

This produces a PSpice netlist declaration like this:

```
GLOSSY 5 0 LAPLACE {V(10)} = {exp(-sqrt(C*s*(R + L*s)))}
```

The Laplace transform parts are, however, an inefficient way, in both computer time and memory, to implement a delay. For ideal delays we recommend using the transmission line part instead.

Math functions

The ABM math function parts are shown in [Table 1](#). For each device, the corresponding template is shown, indicating the order in which the inputs are processed, if applicable.

Table 1 *ABM math function parts*

For this device...	Output is the...
ABS	absolute value of the input
SQRT	square root of the input
PWR	result of raising the absolute value of the input to the power specified by EXP
PWRS	result of raising the (signed) input value to the power specified by EXP
LOG	LOG of the input
LOG10	LOG ₁₀ of the input
EXP	result of e raised to the power specified by the input value (e^x where x is the input)
SIN	\sin of the input (where the input is in radians)
COS	\cos of the input (where the input is in radians)
TAN	\tan of the input (where the input is in radians)
ATAN, ARCTAN	\tan^{-1} of the input (where the output is in radians)

Math function parts are based on the PSpice “E” device type. Each provides one or more inputs, and a mathematical function which is applied to the input. The result is output on the output net.

ABM expression parts

The expression parts are shown in [Table 2](#). These parts can be customized to perform a variety of functions depending on your requirements. Each of these parts has a set of four expression *building block* properties of the form:

$$\text{EXP}n$$

where $n = 1, 2, 3$, or 4 .

During netlist generation, the complete expression is formed by concatenating the building block expressions in numeric order, thus defining the transfer function. Hence, the first expression fragment should be assigned to the EXP1 property, the second fragment to EXP2, and so on.

Expression properties can be defined using a combination of arithmetic operators and input designators. You may use any of the standard PSpice arithmetic operators (see [Table 9 on page 3-70](#)) within an expression statement. You may also use the EXP n properties as variables to represent nets or constants.

Table 2 ABM expression parts

Part	Inputs	Output
ABM	none	V
ABM1	1	V
ABM2	2	V
ABM3	3	V
ABM/I	none	I
ABM1/I	1	I
ABM2/I	2	I
ABM3/I	3	I

The following examples illustrate a variety of ABM expression part applications.

Example one

Suppose you want to set an output voltage on net 4 to 5 volts times the square root of the voltage between nets 3 and 2. You could use an ABM2 part (which takes two inputs and provides a voltage output) to define a part like the one shown in Figure 43.

In this example of an ABM device, the output voltage is set to 5 volts times the square root of the voltage between net 3 and net 2. The property settings for this part are as follows:

```
EXP1 = 5V *
EXP2 = Sqrt(V(%IN2,%IN1))
```

This will produce a PSpice netlist declaration like this:

```
ESQROOT 4 0 VALUE = {5V*Sqrt(V(3,2))}
```

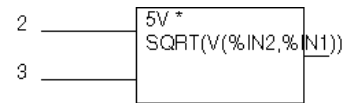


Figure 43 ABM expression part example one.

Example two

GPSK is an oscillator for a PSK (Phase Shift Keyed) modulator. Current is pumped from net 11 through the source to net 6. Its value is a sine wave with an amplitude of 15 mA and a frequency of 10 kHz. The voltage at net 3 can shift the phase of GPSK by 1 radian/volt. Note the use of the TIME parameter in the EXP2 expression. This is the PSpice internal sweep variable used in transient analyses. For any analysis other than transient, TIME = 0. This could be represented with an ABM1/I part (single input, current output) like the one shown in Figure 44.

This part is characterized by the following properties:

```
EXP1 = 15ma * SIN(
EXP2 = 6.28*10kHz*TIME
EXP3 = + V(%IN))
```

This produces a PSpice netlist declaration like this:

```
GPSK 11 6 VALUE = {15MA*SIN(6.28*10kHz*TIME+V(3))}
```

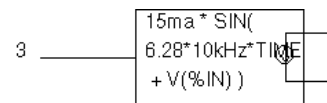


Figure 44 ABM expression part example two.

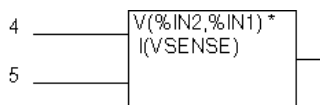


Figure 45 ABM expression part example three.

Example three

A device, EPWR, computes the instantaneous power by multiplying the voltage across nets 5 and 4 by the current through VSENSE. Sources are controlled by expressions which may contain voltages or currents or both. The ABM2 part (two inputs, current output) in Figure 45 could represent this.

This part is characterized by the following properties:

$$\begin{aligned} \text{EXP1} &= V(\%IN2, \%IN1) * \\ \text{EXP2} &= I(VSENSE) \end{aligned}$$

This produces a PSpice netlist declaration like this:

```
EPWR 3 0 VALUE = {V(5,4)*I(VSENSE)}
```

Example four

The output of a component, GRATIO, is a current whose value (in amps) is equal to the ratio of the voltages at nets 13 and 2. If $V(2) = 0$, the output depends upon $V(13)$ as follows:

$$\begin{aligned} \text{if } V(13) &= 0, \text{ output} = 0 \\ \text{if } V(13) &> 0, \text{ output} = \text{MAXREAL} \\ \text{if } V(13) &< 0, \text{ output} = -\text{MAXREAL} \end{aligned}$$

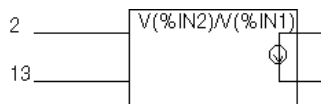


Figure 46 ABM expression part example four.

where MAXREAL is a PSpice internal constant representing a very large number (on the order of $1e30$). In general, the result of evaluating an expression is limited to MAXREAL. This is modeled with an ABM2/I (two input, current output) part like this one in Figure 46.

This part is characterized by the following properties:

$$\text{EXP1} = V(\%IN2)/V(\%IN1)$$

Note that output of GRATIO can be used as part of the controlling function. This produces a PSpice netlist declaration like this:

```
GRATIO 2 3 VALUE = {V(13)/V(2)}
```

Note *Letting a current approach $\pm 1e30$ will almost certainly cause convergence problems. To avoid this, use the limit function on the ratio to keep the current within reasonable limits.*

An instantaneous device example: modeling a triode

This section provides an example of using various ABM parts to model a triode vacuum tube. The schematic of the triode subcircuit is shown in Figure 47.

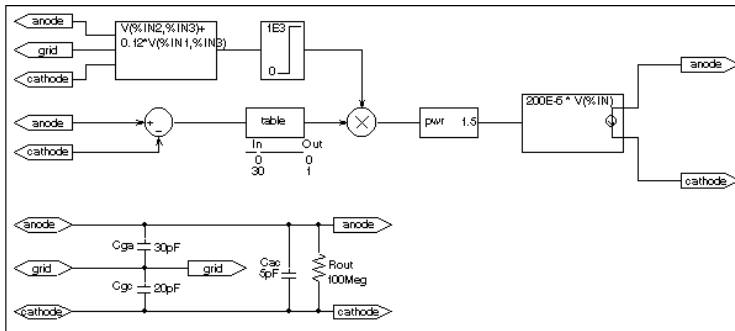


Figure 47 *Triode circuit.*

Assumptions: In its main operating region, the triode's current is proportional to the $3/2$ power of a linear combination of the grid and anode voltages:

$$i_{\text{anode}} = k_0 \cdot (v_g + k_1 \cdot v_a)^{1.5}$$

For a typical triode, $k_0 = 200\text{e-}6$ and $k_1 = 0.12$.

Looking at the upper left-hand portion of the schematic, notice the a general-purpose ABM part used to take the input voltages from anode, grid, and cathode. Assume the following associations:

- $V(\text{anode})$ is associated with $V(\%IN1)$
- $V(\text{grid})$ is associated with $V(\%IN2)$
- $V(\text{cathode})$ is associated with $V(\%IN3)$

The expression property EXP1 then represents $V(\text{grid}, \text{cathode})$ and the expression property EXP2 represents $0.12[V(\text{anode}, \text{cathode})]$. When the template substitution is performed, the resulting VALUE is equivalent to the following:

$$V = V(\text{grid}, \text{cathode}) + 0.12 \cdot V(\text{anode}, \text{cathode})$$

The part would be defined with the following characteristics:

$$\begin{aligned}\text{EXP1} &= V(\%IN2, \%IN3) + \\ \text{EXP2} &= 0.12 * V(\%IN1, \%IN3)\end{aligned}$$

This works for the main operating region but does not model the case in which the current stays 0 when combined grid and anode voltages go negative. We can accommodate that situation as follows by adding the LIMIT part with the following characteristics:

$$\begin{aligned}\text{HI} &= 1\text{E}3 \\ \text{LO} &= 0\end{aligned}$$

This part instance, LIMIT1, converts all negative values of $v_g + 0.12 * v_a$ to 0 and leaves all positive values (up to 1 kV) alone. For a more realistic model, we could have used TABLE to correctly model how the tube turns off at 0 or at small negative grid voltages.

We also need to make sure that the current becomes zero when the anode alone goes negative. To do this, we can use a DIFF device, (immediately below the ABM3 device) to monitor the difference between V(anode) and V(cathode), and output the difference to the TABLE part. The table translates all values at or below zero to zero, and all values greater than or equal to 30 to one. All values between 0 and 30 are linearly interpolated. The properties for the TABLE part are as follows:

$$\begin{aligned}\text{ROW1} &= 00 \\ \text{ROW2} &= 301\end{aligned}$$

The TABLE part is a simple one, and ensures that only a zero value is output to the multiplier for negative anode voltages.

The output from the TABLE part and the LIMIT part are combined at the MULT multiplier part. The output of the MULT part is the product of the two input voltages. This value is then raised to the 3/2 or 1.5 power using the PWR part. The exponential property of the PWR part is defined as follows:

$$\text{EXP} = 1.5$$

The last major component is an ABM expression component to take an input voltage and convert it into a current. The relevant ABM1/I part property looks like this:

$$\text{EXP1} = 200\text{E-6} * \text{V}(\% \text{IN})$$

A final step in the model is to add device parasitics. For example, a resistor can be used to give a finite output impedance. Capacitances between the grid, cathode, and anode are also needed. The lower part of the schematic in Figure 47 shows a possible method for incorporating these effects. To complete the example, one could add a circuit which produces the family of I-V curves (shown in Figure 48).

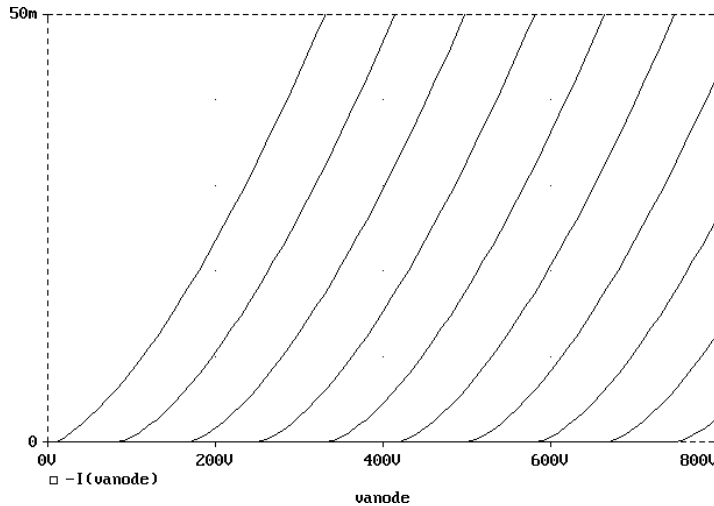


Figure 48 Triode subcircuit producing a family of I-V curves.

PSpice-equivalent parts

PSpice-equivalent parts respond to a differential input and have double-ended output. These parts reflect the structure of PSpiceE and G devices, thus having two pins for each controlling input and the output in the part. [Table 1](#) summarizes the PSpice-equivalent parts available in the part library.

Table 1 *PSpice-equivalent parts*

Category	Part	Description	Properties
Mathematical expression	EVALUE	general purpose	EXPR
	GVALUE		
	ESUM	special purpose	(none)
	GSUM		
	EMULT		
	GMULT		
Table look-up	ETABLE	general purpose	EXPR
	GTABLE		TABLE
Frequency table look-up	EFREQ	general purpose	EXPR
	GFREQ		TABLE
Laplace transform	ELAPLACE	general purpose	EXPR
	GLAPLACE		XFORM

There are no equivalent “F” or “H” part types in the part library because PSpice “F” and “H” devices do not support the ABM extensions.

PSpice-equivalent ABM parts can be classified as either E or G device types. The E part type provides a voltage output, and the G device type provides a current output. The device’s transfer function can contain any mixture of voltages and currents as inputs. Hence, there is no longer a division between voltage-controlled and current-controlled parts. Rather the part type is dictated only by the output requirements. If a voltage output is required, use an E part type. If a current output is necessary, use a G part type.

Each E or G part type in the ABM.OLB part file is defined by a template that provides the specifics of the transfer function. Other properties in the model definition can be edited to customize the transfer function. By default, the template cannot be modified directly choosing Properties from the Edit menu in Capture. Rather, the values for other properties (such as the expressions used in the template) are usually edited, then these values are substituted into the template. However, the part editor can be used to modify the template or designate the template as modifiable from within Capture. This way, custom parts can be created for special-purpose application.

Implementation of PSpice-equivalent parts

Although you generally use Capture to place and specify PSpice-equivalent ABM parts, it is useful to know the PSpice command syntax for “E” and “G” devices. This is especially true when creating custom ABM parts since part templates must adhere to PSpice syntax.

The general forms for PSpice “E” and “G” extensions are:

```
E <name> <connecting nodes> <ABM keyword> <ABM function>
G <name> <connecting nodes> <ABM keyword> <ABM function>
```

where

<i><name></i>	is the device name appended to the E or G device type character
<i><connecting nodes></i>	specifies the <i><(+ node name, - node name)></i> pair between which the device is connected

<i><ABM keyword></i>	specifies the form of the transfer function to be used, as one of: VALUE arithmetic expression TABLE lookup table LAPLACE Laplace transform FREQ frequency response table CHEBYSHEV Chebyshev filter characteristics
<i><ABM function></i>	specifies the transfer function as a formula or lookup table as required by the specified <i><ABM keyword></i>

Refer to the online *OrCAD PSpice A/D Reference Manual* for detailed information.

Modeling mathematical or instantaneous relationships

The instantaneous models (using VALUE and TABLE extensions to PSpice “E” and “G” devices in the part templates) enforce a direct response to the input at each moment in time. For example, the output might be equal to the square root of the input at every point in time. Such a device has no memory, or, a flat frequency response. These techniques can be used to model both linear and nonlinear responses.

Note *For AC analysis, a nonlinear device is first linearized around the bias point, and then the linear equivalent is used.*

EVALUE and GVALUE parts

The EVALUE and GVALUE parts allow an instantaneous transfer function to be written as a mathematical expression in standard notation. These parts take the input signal, perform the function specified by the EXPR property on the signal, and output the result on the output pins.

In controlled sources, EXPR may contain constants and parameters as well as voltages, currents, or time. Voltages may be either the voltage at a net, such as V(5), or the voltage across two nets, such as V(4,5). Currents must be the current through a voltage source (V device), for example, I(VSENSE). Voltage sources with a value of 0 are handy for sensing current for use in these expressions.

Functions may be used in expressions, along with arithmetic operators (+, -, *, and /) and parentheses. Available built-in functions are summarized in [Table 10 on page 3-71](#).

The EVALUE and GVALUE parts are defined, in part, by the following properties (default values are shown):

EVALUE

EXPR V(%IN+, %IN-)

GVALUE

EXPR V(%IN+, %IN-)

Sources are controlled by expressions which may contain voltages, currents, or both. The following examples illustrate customized EVALUE and GVALUE parts.

Example 1

In the example of an EVALUE device shown in Figure 49, the output voltage is set to 5 volts times the square root of the voltage between pins %IN+ and %IN-.

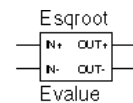
The property settings for this device are as follows:

EXPR = 5v * SQRT(V(%IN+, %IN-))

Example 2

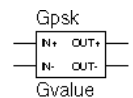
Consider the device in Figure 50. This device could be used as an oscillator for a PSK (Phase Shift Keyed) modulator.

A current through a source is a sine wave with an amplitude of 15 mA and a frequency of 10 kHz. The voltage at the input pin can shift the phase by 1 radian/volt. Note the use of the TIME parameter in this



5v * SQRT(V(%IN+, %IN-))

Figure 49 EVALUE part example.



15ma*SIN(6.28*10kHz*TIME+V(%IN+, %IN-))

Figure 50 GVALUE part example.

expression. This is the PSpice internal sweep variable used in transient analyses. For any analysis other than transient, $TIME = 0$. The relevant property settings for this device are shown below:

```
EXPR = 15ma*SIN(6.28*10kHz*TIME+V(%IN+,%IN-))
```

EMULT, GMULT, ESUM, and GSUM

The EMULT and GMULT parts provide output which is based on the product of two input sources. The ESUM and GSUM parts provide output which is based on the sum of two input sources. The complete transfer function may also include other mathematical expressions.

Example 1

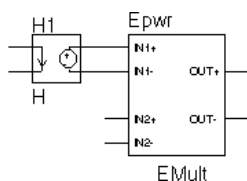


Figure 51 EMULT part example.

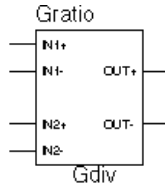
Consider the device in Figure 51. This device computes the instantaneous power by multiplying the voltage across pins %IN+ and %IN- by the current through VSENSE. This device's behavior is built-in to the PSPICETEMPLATE property as follows (appears on one line):

```
TEMPLATE=E^@REFDES %OUT+ %OUT- VALUE {V(%IN1+,%IN1-)  
*V(%IN2+,%IN2-)}
```

You can use the part editor to change the characteristics of the template to accommodate additional mathematical functions, or to change the nature of the transfer function itself. For example, you may want to create a voltage divider, rather than a multiplier. This is illustrated in the following example.

Example 2

Consider the device in Figure 52.



```
G^@REFDES %OUT+ %OUT- VALUE {V(%IN1+,%IN1-)/V(%IN2+,%IN2-)}
```

Figure 52 *GMULT part example.*

With this device, the output is a current is equal to the ratio of the voltages at input pins 1 and input pins 2. If $V(\%IN2+, \%IN2-) = 0$, the output depends upon $V(\%IN1+, \%IN1-)$ as follows:

- if $V(\%IN1+, \%IN1-) = 0$, output = 0
- if $V(\%IN1+, \%IN1-) > 0$, output = MAXREAL
- if $V(\%IN1+, \%IN1-) < 0$, output = -MAXREAL

where MAXREAL is a PSpice internal constant representing a very large number (on the order of $1e30$). In general, the result of evaluating an expression is limited to MAXREAL. Note that the output of the part can also be used as part of the controlling function.

To create this device, you would first make a new part, GDIV, based on the GMULT part. Edit the GDIV template to divide the two input values rather than multiply them.

Lookup tables (ETABLE and GTABLE)

The ETABLE and GTABLE parts use a transfer function described by a table. These device models are well suited for use with measured data.

The ETABLE and GTABLE parts are defined in part by the following properties (default values are shown):

ETABLE

TABLE	(-15, -15), (15,15)
EXPR	$V(\%IN+, \%IN-)$

GTABLE

TABLE	(-15, -15), (15,15)
EXPR	V(%IN+, %IN-)

First, EXPR is evaluated, and that value is used to look up an entry in the table. EXPR is a function of the input (current or voltage) and follows the same rules as for VALUE expressions.

The table consists of pairs of values, the first of which is an input, and the second of which is the corresponding output. Linear interpolation is performed between entries. For values of EXPR outside the table's range, the device's output is a constant with a value equal to the entry with the smallest (or largest) input. This characteristic can be used to impose an upper and lower limit on the output.

An example of a table declaration (using the TABLE property) would be the following:

```
TABLE =
+ (0, 0) (.02, 2.690E-03) (.04, 4.102E-03) (.06, 4.621E-03)
+ (.08, 4.460E-03) (.10, 3.860E-03) (.12, 3.079E-03) (.14,
+ 2.327E-03)
+ (.16, 1.726E-03) (.18, 1.308E-03) (.20, 1.042E-03) (.22,
+ 8.734E-04)
+ (.24, 7.544E-04) (.26, 6.566E-04) (.28, 5.718E-04) (.30,
+ 5.013E-04)
+ (.32, 4.464E-04) (.34, 4.053E-04) (.36, 3.781E-04) (.38,
+ 3.744E-04)
+ (.40, 4.127E-04) (.42, 5.053E-04) (.44, 6.380E-04) (.46,
+ 7.935E-04)
+ (.48, 1.139E-03) (.50, 2.605E-03) (.52, 8.259E-03) (.54,
+ 2.609E-02)
+ (.56, 7.418E-02) (.58, 1.895E-01) (.60, 4.426E-01)
```

Frequency-domain device models

Frequency-domain models (ELAPLACE, GLAPLACE, EFREQ, and GFREQ) are characterized by output that depends on the current input as well as the input history. The relationship is therefore non-instantaneous. For example, the output may be equal to the integral of the input over time. In other words, the response depends upon frequency.

During AC analysis, the frequency response determines the complex gain at each frequency. During DC analysis and bias point calculation, the gain is the zero-frequency response. During transient analysis, the output of the device is the convolution of the input and the impulse response of the device.

Laplace transforms (LAPLACE)

The ELAPLACE and GLAPLACE parts allow a transfer function to be described by a Laplace transform function. The ELAPLACE and GLAPLACE parts are defined, in part, by the following properties (default values are shown):

ELAPLACE

EXPR	V(%IN+, %IN-)
XFORM	1/s

GLAPLACE

EXPR	V(%IN+, %IN-)
XFORM	1/s

The LAPLACE parts use a Laplace transform description. The input to the transform is the value of EXPR, where EXPR follows the same rules as for VALUE expressions (see [EVALUE](#) and [GVALUE](#) parts on page 6-176). XFORM is an expression in the Laplace variable, s. It follows the rules for standard expressions as described for VALUE expressions with the addition of the s variable.

Moving back and forth between the time and frequency-domains can cause surprising results. Often the results are quite different than what one would intuitively expect. For this reason, we strongly recommend familiarity with a reference on Fourier and Laplace transforms. A good one is:

- 1 R. Bracewell, *The Fourier Transform and Its Applications*, McGraw-Hill, Revised Second Edition (1986)

We also recommend familiarity with the use of transforms in analyzing linear systems. Some references on this subject:

- 2 W. H. Chen, *The Analysis of Linear Systems*, McGraw-Hill (1962)
- 3 J. A. Aseltine, *Transform Method in Linear System Analysis*, McGraw-Hill (1958)
- 4 G. R. Cooper and C. D. McGillen, *Methods of Signal and System Analysis*, Holt, Rinehart, and Winston (1967)

Voltages, currents, and TIME cannot appear in a Laplace transform.

The output of the device depends on the type of analysis being done. For DC and bias point, the output is simply the zero frequency gain times the value of EXPR. The zero frequency gain is the value of XFORM with $s = 0$. For AC analysis, EXPR is linearized around the bias point (similar to the VALUE parts). The output is then the input times the gain of EXPR times the value of XFORM. The value of XFORM at a frequency is calculated by substituting $j\omega$ for s , where ω is 2 π -frequency. For transient analysis, the value of EXPR is evaluated at each time point. The output is then the convolution of the past values of EXPR with the impulse response of XFORM. These rules follow the standard method of using Laplace transforms. We recommend looking at one or more of the references cited in [Frequency-domain device models on page 6-181](#) for more information.

Example

The input to the Laplace transform is the voltage across the input pins, or $V(\%IN+, \%IN-)$. The EXPR property may be edited to include constants or functions, as with other parts. The transform, $1/(1+.001\cdot s)$, describes a simple, lossy integrator with a time constant of 1 millisecond. This can be implemented with an RC pair that has a time constant of 1 millisecond.

Using the part editor, you would define the XFORM and EXPR properties as follows:

```
XFORM = 1/(1+.001*s)
EXPR = V(%IN+, %IN-)
```

The default template remains (appears on one line):

```
TEMPLATE= E^@REFDES %OUT+ %OUT- LAPLACE
{@EXPR}= (@XFORM)
```

After netlist substitution of the template, the resulting transfer function would become:

```
V(%OUT+, %OUT-) = LAPLACE {V(%IN+, %IN-)} = (1/(1+.001*s))
```

The output is a voltage and is applied between pins %OUT+ and %OUT-. For DC, the output is simply equal to the input, since the gain at $s = 0$ is 1.

For AC analysis, the gain is found by substituting $j\omega$ for s . This gives a flat response out to a corner frequency of $1000/(2\pi) = 159$ Hz and a roll-off of 6 dB per octave after 159 Hz. There is also a phase shift centered around 159 Hz. In other words, the gain has both a real and an imaginary component. The gain and phase characteristic is the same as that shown for the equivalent control system part example using the LAPLACE part (see Figure 41 on page 6-165).

For transient analysis, the output is the convolution of the input waveform with the impulse response of $1/(1+.001\cdot s)$. The impulse response is a decaying exponential with a time constant of 1 millisecond. This means that the output is the “lossy integral” of the input, where the loss has a time constant of 1 millisecond.

This will produce a PSpice netlist declaration similar to:

```
ERC 5 0 LAPLACE {V(10)} = {1/(1+.001*s)}
```

Frequency response tables (EFREQ and GFREQ)

The EFREQ and GFREQ parts are described by a table of frequency responses in either the magnitude/phase domain or complex number domain. The entire table is read in and converted to magnitude in dB and phase in degrees. Interpolation is performed between entries. Phase is interpolated linearly; magnitude is interpolated logarithmically. For frequencies outside the table's range, 0 (zero) magnitude is used.

EFREQ and GFREQ properties are defined as follows:

EXPR	value used for table lookup; defaults to V(%IN+, %IN-) if left blank.
TABLE	series of either (input frequency, magnitude, phase) triplets, or (input frequency, real part, imaginary part) triplets describing a complex value; defaults to (0,0,0) (1Meg,-10,90) if left blank.

DELAY	group delay increment; defaults to 0 if left blank.
R_I	table type; if left blank, the frequency table is interpreted in the (input frequency, magnitude, phase) format; if defined with any value (such as YES), the table is interpreted in the (input frequency, real part, imaginary part) format.
MAGUNITS	units for magnitude where the value can be DB (decibels) or MAG (raw magnitude); defaults to DB if left blank.
PHASEUNITS	units for phase where the value can be DEG (degrees) or RAD (radians); defaults to DEG if left blank.

The DELAY property increases the group delay of the frequency table by the specified amount. The delay term is particularly useful when an EFREQ or GFREQ device generates a non-causality warning message during a transient analysis. The warning message issues a delay value that can be assigned to the part's DELAY property for subsequent runs, without otherwise altering the table.

The output of the device depends on the analysis being done. For DC and bias point, the output is simply the zero frequency magnitude times the value of EXPR. For AC analysis, EXPR is linearized around the bias point (similar to EVALUE and GVALUE parts). The output for each frequency is then the input times the gain of EXPR times the value of the table at that frequency. For transient analysis, the value of EXPR is evaluated at each time point. The output is then the convolution of the past values of EXPR with the impulse response of the frequency response. These rules follow the standard method of using Fourier transforms. We recommend looking at one or more of the references cited in [Frequency-domain device models on page 6-181](#) for more information.

Note *The table's frequencies must be in order from lowest to highest.*

Figure 53 shows an EFREQ device used as a low pass filter. The input to the frequency response is the voltage across the input pins. The table describes a low pass filter with a response of 1 (0 dB) for frequencies below 5 kilohertz and a response of .001 (-60 dB) for frequencies above 6 kilohertz. The output is a voltage across the output pins.

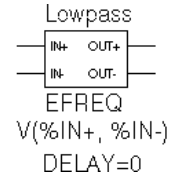


Figure 53 EFREQ part example.

This part is defined by the following properties:

```
TABLE = (0, 0, 0) (5kHz, 0, -5760) (6kHz, -60, -6912)
DELAY =
R_I =
MAGUNITS =
PHASEUNITS =
```

Since R_I, MAGUNITS, and PHASEUNITS are undefined, each table entry is interpreted as containing frequency, magnitude value in dB, and phase values in degrees. Delay defaults to 0.

The phase lags linearly with frequency meaning that this table exhibits a constant time (group) delay. The delay is necessary so that the impulse response is causal. That is, so that the impulse response does not have any significant components before time zero.

The constant group delay is calculated from the values for a given table entry as follows:

$$\text{group delay} = \text{phase} / 360 / \text{frequency}$$

For this example, the group delay is 3.2 msec ($6912 / 360 / 6\text{k} = 5760 / 360 / 6\text{k} = 3.2\text{m}$). An alternative specification for this table could be:

```
TABLE = (0, 0, 0) (5kHz, 0, 0) (6kHz, -60, 0)
DELAY = 3.2ms
R_I =
MAGUNITS =
PHASEUNITS =
```

This produces a PSpice netlist declaration like this:

```
ELOWPASS 5 0 FREQ {V(10)} = (0,0,0) (5kHz,0,0) (6kHz-60,0)
+ DELAY = 3.2ms
```

Cautions and recommendations for simulation and analysis

Instantaneous device modeling

During AC analysis, nonlinear transfer functions are handled the same way as other nonlinear parts: each function is linearized around the bias point and the resulting small-signal equivalent is used.

Consider the voltage multiplier (mixer) shown in Figure 54. This circuit has the following characteristics:

Vin1: DC=0v AC=1v
Vin2: DC=0v AC=1v

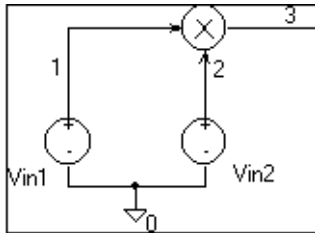


Figure 54 Voltage multiplier circuit (mixer).

where the output on net 3 is $V(1)*V(2)$.

During AC analysis, $V(3) = 0$ due to the 0 volts bias point voltage on nets 1, 2, and 3. The small-signal equivalent therefore has 0 gain (the derivative of $V(1)*V(2)$ with respect to both $V(1)$ and $V(2)$ is 0 when $V(1)=V(2)=0$). So, the output of the mixer during AC analysis will be 0 regardless of the AC values of $V(1)$ and $V(2)$.

Another way of looking at this is that a mixer is a nonlinear device and AC analysis is a linear analysis. The output of the mixer has 0 amplitude at the fundamental. (Output is nonzero at DC and twice the input frequency, but these are not included in a linear analysis.)

If you need to analyze nonlinear functions, such as a mixer, use transient analysis. Transient analysis solves the full, nonlinear circuit equations. It also allows you to use input waveforms with different frequencies (for example, VIN1 could be 90 MHz and VIN2 could be 89.8 MHz). AC analysis does not have this flexibility, but in return it uses much less computer time.

Frequency-domain parts

Some caution is in order when moving between frequency and time domains. This section discusses several points that are involved in the implementation of frequency-domain parts. These discussions all involve the transient analysis, since both the DC and AC analyses are straightforward.

The first point is that there are limits on the maximum values and on the resolution of both time and frequency. These are related: the frequency resolution is the inverse of the maximum time and vice versa. The maximum time is the length of the transient analysis, TSTOP. Therefore, the frequency resolution is $1/TSTOP$.

Laplace transforms

For Laplace transforms, PSpice starts off with initial bounds on the frequency resolution and the maximum frequency determined by the transient analysis parameters as follows. The frequency resolution is initially set below the theoretical limit to $(.25/TSTOP)$ and is then made as large as possible without inducing sampling errors. The maximum frequency has an initial upper bound of $(1/(RELTOL*TMAX))$, where TMAX is the transient analysis Step Ceiling value, and RELTOL is the relative accuracy of all calculated voltages and currents. If a Step Ceiling value is not specified, PSpice uses the Transient Analysis Print Step, TSTEP, instead.

PSpice then attempts to reduce the maximum frequency by searching for the frequency at which the response has fallen to RELTOL times the maximum response. For instance, for the transform:

$$1/(1+s)$$

the maximum response, 1.0, is at $s = j\omega = 0$ (DC). The cutoff frequency used when $RELTOL=.001$, is approximately $1000/(2\pi) = 159$ Hz. At 159 Hz, the response is down to .001 (down by 60 db). Since some

Note TSTOP, TMAX, and TSTEP values are configured using Transient on the Setup menu. The RELTOL property is set using Options on the Setup menu.

transforms do not have such a limit, there is also a limit of $10/\text{RELTOL}$ times the frequency resolution, or $10/(\text{RELTOL} \cdot \text{TSTOP})$. For example, consider the transform:

$$e^{-0.001 \cdot s}$$

This is an ideal delay of 1 millisecond and has no frequency cutoff. If $\text{TSTOP} = 10$ milliseconds and $\text{RELTOL} = .001$, then PSpice imposes a frequency cutoff of 10 MHz. Since the time resolution is the inverse of the maximum frequency, this is equivalent to saying that the delay cannot resolve changes in the input at a rate faster than .1 microseconds. In general, the time resolution will be limited to $\text{RELTOL} \cdot \text{TSTOP} / 10$.

A final computational consideration for Laplace parts is that the impulse response is determined by means of an FFT on the Laplace expression. The FFT is limited to 8192 points to keep it tractable, and this places an additional limit on the maximum frequency, which may not be greater than 8192 times the frequency resolution.

If your circuit contains many Laplace parts which can be combined into a more complex single device, it is generally preferable to do this. This saves computation and memory since there are fewer impulse responses. It also reduces the number of opportunities for numerical artifacts that might reduce the accuracy of your transient analyses.

Laplace transforms can contain poles in the left half-plane. Such poles will cause an impulse response that increases with time instead of decaying. Since the transient analysis is always for a finite time span, PSpice does not have a problem calculating the transient (or DC) response. However, such poles will make the actual device oscillate.

Non-causality and Laplace transforms

PSpice applies an inverse FFT to the Laplace expression to obtain an impulse response, and then convolves the impulse response with the dependent source input to obtain the output. Some common impulse responses are inherently non-causal. This means that the convolution

must be applied to both past and future samples of the input in order to properly represent the inverse of the Laplace expression.

For example, the expression $\{S\}$ corresponds to differentiation in the time domain. The impulse response for $\{S\}$ is an impulse pair separated by an infinitesimal distance in time. The impulses have opposite signs, and are situated one in the infinitesimal past, the other in the infinitesimal future. In other words, convolution with this corresponds to applying a finite-divided difference in the time domain.

The problem with this for PSpice is that the simulator only has the present and past values of the simulated input, so it can only apply half of the impulse pair during convolution. This will obviously not result in time-domain differentiation. PSpice can detect, but not fix this condition, and issues a non-causality warning message when it occurs. The message tells what percentage of the impulse response is non-causal, and how much delay would need to be added to slide the non-causal part into a causal region. $\{S\}$ is theoretically 50% non-causal. Non-causality on the order of 1% or less is usually not critical to the simulation results.

You can delay $\{S\}$ to keep it causal, but the separation between the impulses is infinitesimal. This means that a very small time step is needed. For this reason, it is usually better to use a macromodel to implement differentiation.

Here are some guidelines:

- In the case of a Laplace device (ELAPLACE), multiply the Laplace expression by e to the $(-s * \text{the suggested delay})$.
- In the case of a frequency table (EFREQ or GFREQ), do either of the following:
 - Specify the table with `DELAY=<the suggested delay>`.
 - Compute the delay by adding a phase shift.

Chebyshev filters

All of the considerations given above for Laplace parts also apply to Chebyshev filter parts. However, PSpice also attempts to deal directly with inaccuracies due to sampling by applying Nyquist criteria based on the highest filter cutoff frequency. This is done by checking the value of TMAX. If TMAX is not specified it is assigned a value, or if it is specified, it may be reduced.

For low pass and band pass filters, TMAX is set to $(0.5/FS)$, where FS is the stop band cutoff in the case of a low pass filter, or the upper stop band cutoff in the case of a band pass filter.

For high pass and band reject filters, there is no clear way to apply the Nyquist criterion directly, so an additional factor of two is thrown in as a safety margin. Thus, TMAX is set to $(0.25/FP)$, where FP is the pass band cutoff for the high pass case or the upper pass band cutoff for the band reject case. It may be necessary to set TMAX to something smaller if the filter input has significant frequency content above these limits.

Frequency tables

For frequency response tables, the maximum frequency is twice the highest value. It will be reduced to $10/(RELTOL \cdot TSTOP)$ or 8192 times the frequency resolution if either value is smaller.

The frequency resolution for frequency response tables is taken to be either the smallest frequency increment in the table or the fastest rate of phase change, whichever is least. PSpice then checks to see if it can be loosened without inducing sampling errors.

Trading off computer resources for accuracy

There is a significant trade-off between accuracy and computation time for parts modeled in the frequency domain. The amount of computer time and memory scale approximately inversely to RELTOL. Therefore, if you can use RELTOL=.01 instead of the default .001, you will be ahead. However, this will not adversely affect the impulse response. You may also wish to vary TMAX and TSTOP, since these also come into play.

Since the trade-off issues are fairly complex, it is advisable to first simulate a small test circuit containing only the frequency-domain device, and then after proper validation, proceed to incorporate it in your larger design. The PSpice defaults will be appropriate most of the time if accuracy is your main concern, but it is still worth checking.

Note *Do not set RELTOL to a value above 0.01. This can seriously compromise the accuracy of your simulation.*

Basic controlled sources

As with basic SPICE, PSpice has basic controlled sources derived from the standard SPICE E, F, G, and H devices. [Table 1](#) summarizes the linear controlled source types provided in the standard part library.

Table 1 Basic controlled sources in ANALOG.OLB

Device type	Part name
Controlled Voltage Source (PSpice E device)	E
Current-Controlled Current Source (PSpice F device)	F
Controlled Current Source (PSpice G device)	G
Current-Controlled Voltage Source (PSpice H device)	H

Refer to your *OrCAD Capture User's Guide* for a description of how to create a custom part.

Creating custom ABM parts

Create a custom part when you need a controlled source that is not provided in the special purpose set or that is more elaborate than you can build with the general purpose parts (with multiple controlling inputs, for example).

The transfer function can be built into the part two different ways:

- directly in the PSPICETEMPLATE definition.
- by defining the part's EXPR and related properties (if any).

Refer to the online *OrCAD PSpice A/D Reference Manual* for more information about E and G devices.

The PSpice syntax for declaring E and G devices can help you form a PSPICETEMPLATE definition.

Part three

Setting Up and Running Analyses

Part Three describes how to set up and run analyses and provides setup information specific to each analysis type.

- [Chapter 7, Setting up analyses and starting simulation](#), explains the procedures general to all analysis types to set up and start the simulation.
- [Chapter 8, DC analyses](#), describes how to set up DC analyses, including DC sweep, bias point detail, small-signal DC transfer, and DC sensitivity.
- [Chapter 9, AC analyses](#), describes how to set up AC sweep and noise analyses.
- [Chapter 10, Transient analysis](#), describes how to set up transient analysis and optionally Fourier components. This chapter also explains how to use the Stimulus Editor to create time-based input.
- [Chapter 11, Parametric and temperature analysis](#), describes how to set up parametric and temperature analyses, and how to run post-simulation performance analysis in Probe on the results of these analyses.

-
- **Chapter 12, Monte Carlo and sensitivity/worst-case analyses**, describes how to set up Monte Carlo and sensitivity/ worst-case analyses for statistical interpretation of your circuit's behavior.

Setting up analyses and starting simulation

7

Chapter overview

This chapter provides an overview of setting up analyses and starting simulation that applies to any analysis type. The other chapters in [Part three, *Setting Up and Running Analyses*](#) provide specific analysis setup information for each analysis type.

This chapter includes the following sections:

- [Analysis types on page 7-196](#)
- [Setting up analyses on page 7-197](#)
- [Starting a simulation on page 7-206](#)

Analysis types

PSpice supports analyses that can simulate analog-only circuits.

Table 2 provides a summary of the available PSpice analyses and the corresponding Analysis type options where the analysis parameters are specified. In Capture, switch to the PSpice view, then from the PSpice menu, choose New Simulation Profile .

Table 2 *Classes of PSpice analyses*

Analysis	Analysis type or Option	Swept variable
Standard analyses		
DC sweep	DC Sweep	source parameter temperature
Bias point	Bias Point	
Small-signal DC transfer	Bias Point	
DC sensitivity	Bias Point	
Frequency response	AC Sweep/Noise	frequency
Noise (requires a frequency response analysis)	AC Sweep/Noise	frequency
Transient response	Time Domain (Transient)	time
Fourier (requires transient response analysis)	Time Domain (Transient)	time
Simple multi-run analyses		
Parametric	Parametric Sweep	
Temperature	Temperature (Sweep)	
Statistical analyses		
Monte Carlo	Monte Carlo/ Worst Case	
Sensitivity/worst-case	Monte Carlo/ Worst Case	

The waveform analyzer calculates and displays the results of PSpice simulations for swept analyses. The waveform analyzer also generates supplementary analysis information in the form of lists and tables, and saves this in the simulation output file.

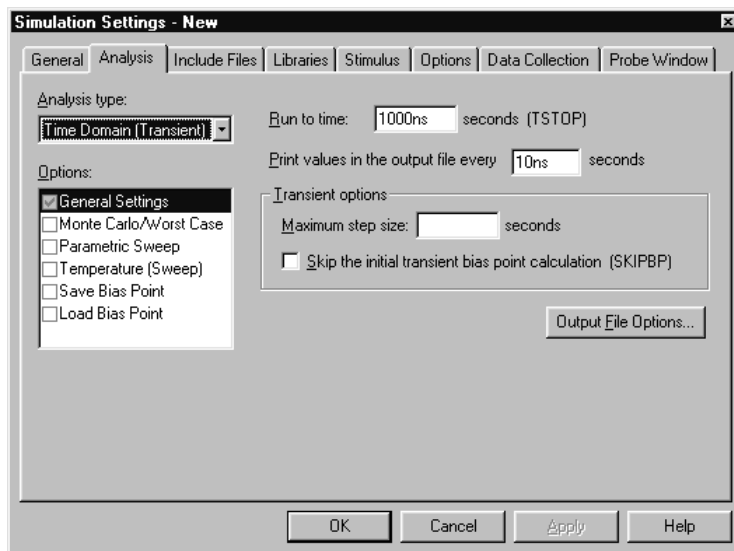
See [Part four, Viewing results](#), for information about using waveform analysis in PSpice.

Setting up analyses

To set up one or more analyses

- 1 From the PSpice menu, choose New Simulation Profile.
- 2 Enter the name of the profile and click OK.
- 3 Click the Analysis tab if it is not already the active tab in the dialog box.

Specific information for setting up each type of analysis is discussed in the following chapters.



- 4 Enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.

See [Output variables on page 7-199](#) for a description of the output variables that can be entered in the Simulation Settings dialog box displayed for an analysis type.

Specific information for setting up each type of analysis is discussed in the following chapters.

- 5 Set up any other analyses you want to perform for the circuit by selecting any of the remaining analysis types and options, then complete their setup dialog boxes.

Execution order for standard analyses

For normal simulations that are run from a simulation profile, or in batch mode, only the particular analysis type that is specified will be executed.

During simulation of a circuit file, the analysis types are performed in the order shown in [Table 3](#). Each type of analysis is conducted only once per run.

Several of the analyses (small-signal transfer, DC sensitivity, and frequency response) depend upon the bias point calculation. Because so many analyses use the bias point, PSpice calculates this automatically. PSpice’s bias point calculation computes initial states of analog components.

Table 3 Execution order for standard analyses

1. DC sweep
2. Bias point
3. Frequency response
4. Noise
5. DC sensitivity
6. Small-signal DC transfer
7. Transient response
8. Fourier components

Output variables

Certain analyses (such as noise, Monte Carlo, sensitivity/worst-case, DC sensitivity, Fourier, and small-signal DC transfer function) require you to specify output variables for voltages and currents at specific points on the schematic. Depending upon the analysis type, you may need to specify the following:

- Voltage on a net, a pin, or at a terminal of a semiconductor device
- Current through a part or into a terminal of a semiconductor device
- A device name

If output variables or other information are required, select Output File Options in the Monte Carlo/Worst Case dialog box and enter the required parameters.

Voltage

Specify voltage in the following format:

$$v[\textit{modifiers}](\textit{<out id>}, \textit{<out id>}) \quad (1)$$

where *<out id>* is:

$$\textit{<net id>} \text{ or } \textit{<pin id>} \quad (2)$$

$$\textit{<net id>} \text{ is a fully qualified net name} \quad (3)$$

$$\textit{<pin id>} \text{ is } \textit{<fully qualified device name>:<pin name>} \quad (4)$$

A fully qualified net name (as referred to in line 3 above) is formed by prefixing the visible net name (from a label applied to one of the segments of a wire or bus, or an offpage port connected to the net) with the full hierarchical path, separated by periods. At the top level of hierarchy, this is just the visible name.

A fully qualified device name (from line 4 above) is distinguished by specifying the full hierarchical path followed by the device's part reference, separated by period characters. For example, a resistor with part reference R34 inside part Y1 placed on a top-level

schematic page is referred to as Y1.R34 when used in an output variable.

A *<pin id>* (from line 4) is uniquely distinguished by specifying the full part name (as described above) followed by a colon, and the pin name. For example, the pins on a capacitor with reference designator C31 placed on a top-level page and pin names 1 and 2 would be identified as C31:1 and C31:2, respectively.

Current

Specify current in the following format:

i[modifiers](<out device>[:modifiers])

where *<out device>* is a fully qualified device name.

Modifiers

The basic syntax for output variables can be modified to indicate terminals of semiconductors and AC specifications. The modifiers come before *<out id>* or *<out device>*. Or, when specifying terminals (such as source or drain), the modifier is the pin name contained in *<out id>*, or is appended to *<out device>* separated by a colon.

Modifiers can be specified as follows:

- For voltage:

v[AC suffix](<out id>[, out id])
v[terminal](<out device>)*

- For current:

i[AC suffix](<out device>[:terminal])
i[terminal][AC suffix](<out device>)

where

terminal specifies one or two terminals for devices with more than two terminals, such as D (drain), G (gate), S (source)

AC suffix specifies the quantity to be reported for an AC analysis, such as M (magnitude), P (phase), G (group delay)

out id specifies either the *<net id>* or *<pin id>* (*<fully qualified device name>:<pin name>*)

out device specifies the *<fully qualified device name>*

These building blocks can be used for specifying output variables as shown in [Table 4](#) (which summarizes the accepted output variable formats) and [Tables 5](#) through [8](#) (which list valid elements for two-terminal, three- or four-terminal devices, transmission line devices, and AC specifications).

Table 4 *PSpice output variable formats*

Format	Meaning
V[ac](< + <i>out id</i> >)	voltage at <i>out id</i>
V[ac](< + <i>out id</i> >,< - <i>out id</i> >)	voltage across + and - <i>out id</i> 's
V[ac](< 2-terminal device <i>out id</i> >)	voltage at a 2-terminal device <i>out id</i>
V[ac](< 3 or 4-terminal device <i>out id</i> >) or V< <i>x</i> >[ac](< 3 or 4-terminal out device >)	voltage at non-grounded terminal <i>x</i> of a 3 or 4-terminal device
V< <i>x</i> >< <i>y</i> >[ac](< 3 or 4-terminal out device >)	voltage across terminals <i>x</i> and <i>y</i> of a 3 or 4-terminal device
V[ac](< transmission line <i>out id</i> >) or V< <i>z</i> >[ac](< transmission line out device >)	voltage at one end <i>z</i> of a transmission line device

Table 4 *PSpice output variable formats (continued)*

Format	Meaning
I[ac](<i>< 3 or 4-terminal out device >: < x ></i>) or I<x>[ac](<i>< 3 or 4-terminal out device ></i>)	current through non-grounded terminal <i>x</i> of a <i>3 or 4-terminal out device</i>
I[ac](<i>< transmission line out device >: < z ></i>) or I<z>[ac](<i>< 3 or 4-terminal out device ></i>)	current through one end <i>z</i> of a <i>transmission line out device</i>
<i>< DC sweep variable ></i>	voltage or current source name

Table 5 *Element definitions for 2-terminal devices*

Device type	<i>< out id > or < out device > device indicator</i>	Output variable examples
capacitor	C	V(CAP:1) I(CAP)
diode	D	V(D23:1) I(D23)
voltage-controlled voltage source	E	V(E14:1) I(E14)
current-controlled current source	F	V(F1:1) I(F1)
voltage-controlled current source	G	V(G2:1) I(G2)
current-controlled voltage source	H	V(HSOURCE:1) I(HSOURCE)
independent current source	I	V(IDRIV:+) I(IDRIV)
inductor	L	V(L1:1) I(L1)

Table 5 *Element definitions for 2-terminal devices*

Device type	<out id> or <out device> device indicator	Output variable examples
resistor	R	V(RC1:1) I(RC1)
voltage-controlled switch	S	V(SWITCH:+) I(SWITCH)
independent voltage source	V	V(VSRC:+) I(VSRC)
current-controlled switch	W	V(W22:-) I(W22)

Table 6 *Element definitions for 3- or 4-terminal devices*

Device type	<out id> or <out device> device indicator	<pin id>	Output variable examples
GaAs MESFET	B	D (Drain terminal)	V(B11:D)
		G (Gate terminal)	ID(B11)
		S (Source terminal)	
Junction FET	J	D (Drain terminal)	VG(JFET)
		G (Gate terminal)	I(JFET:G)
		S (Source terminal)	

Table 6 *Element definitions for 3- or 4-terminal devices*

Device type	<out id> or <out device> device indicator	<pin id>	Output variable examples
MOSFET	M	B (Bulk, substrate terminal) D (Drain terminal) G (Gate terminal) S (Source terminal)	VDG(M1) ID(M1)
bipolar transistor	Q	B (Base terminal) C (Collector terminal) E (Emitter terminal) S (Source terminal)	V(Q1:B) I(Q1:C)
IGBT	Z	C (Collector terminal) E (Emitter terminal) G (Gate terminal)	V(Z1:C) I(Z1:C)

Table 7 *Element definitions for transmission line devices*

Device type	<out id> or <out device> device indicator	<z>	Output variable examples
transmission line	T	A (Port A) B (Port B)	V(T32:A+) I(T32:B-)

Table 8 *Element definitions for AC analysis specific elements*

<i><ac suffix></i> device symbol	Meaning	Output variable examples
(none)	magnitude (default)	V(V1) I(V1)
M	magnitude	VM(CAP1:1) IM(CAP1:1)
DB	magnitude in decibels	VDB(R1)
P	phase	IP(R1)
R	real part	VR(R1)
I	imaginary part	VI(R1)

The INOISE, ONOISE, DB(INOISE), and DB(ONOISE) output variables are predefined for use with noise (AC sweep) analysis.

Starting a simulation

After you have used Capture to enter your circuit design and have set up the analyses to be performed, you can start a simulation by choosing Run from the PSpice menu. When you enter and set up your circuit this way, Capture automatically generates the simulation files and starts PSpice.

There may be situations, however, when you want to run PSpice outside of Capture. You may want to simulate a circuit that was not created in Capture, for example, or you may want to run simulations of multiple circuits in batch mode.

This section includes the following:

- [Starting a simulation from Capture, below](#)
- [Starting a simulation outside of Capture on page 7-207](#)
- [Setting up batch simulations on page 7-207](#)
- [The PSpice simulation window on page 7-208](#)

Starting a simulation from Capture

After you have set up the analyses for the circuit, you can start a simulation from Capture in either of the following ways:

- From the PSpice menu select Run.
- Click the Simulate button on the PSpice toolbar.



Starting a simulation outside of Capture

To start PSpice outside of Capture

- 1 From the Start menu, point to the OrCAD program group, then choose PSpice.
- 2 From the File menu, choose Open Simulation.
- 3 Do one of the following:
 - Double-click on the simulation profile filename (*.SIM) in the list box.
 - Enter the simulation profile filename (*.SIM) in the File name text box and click Open.
- 4 From the Simulation menu, choose Edit Settings to modify any of the analysis setup parameters.
- 5 From the Simulation menu, choose Run (or click the Run toolbar button) to begin the simulation.

Setting up batch simulations

Multiple simulations can be run in batch mode when starting PSpice directly with circuit file input. You can use batch mode, for example, to run a number of simulations overnight. There are two ways to do this, as described below.

Multiple simulation setups within one circuit file

Multiple circuit/simulation descriptions can be concatenated into a single circuit file and simulated all at once with PSpice. Each circuit/simulation description in the file must begin with a title line and end with a .END statement.

The simulator reads all the circuits in the circuit file and then processes each one in sequence. The data file and simulation output file contain the outputs from each circuit in the same order as they appeared in the circuit

file. The effect is the same as if you had run each circuit separately and then concatenated all of the outputs.

Running simulations with multiple circuit files

You can direct PSpice to simulate multiple circuit files using either of the following methods.

Method 1

- 1 From the Start menu, point to the OrCAD program group, then choose PSpice.
- 2 Select Open Simulation from the File menu from the PSpice window.
- 3 Do one of the following:
 - Type each file name in the File Name text box separated by a space.
 - Use the combination keystrokes and mouse clicks in the list box as follows: **Ctrl**+click to select file names one at a time, and **Shift**+click to select groups of files.

Method 2

- 1 From the Start menu, point to the OrCAD program group, then choose PSpice.
- 2 Update the command line in the following way:
 - Include a list of circuit file names separated by spaces.

Circuit file names can be fully qualified or can contain the wild card characters (* and ?).

The PSpice simulation window

The PSpice Simulation Window is an MDI (Multiple Document Interface) application. This implies that you can open and display multiple files at the same time in this

window. For instance, you can have a waveform file (.DAT), a circuit file (.CIR), and a simulation output file (.OUT) open and displayed in different child windows within this one window.

The PSpice Simulation Window consists of three sections: the main window section where the open files are displayed, the output window section where output information such as informational, warning, and error messages from the simulator are shown, and the simulation status window section where detailed status information about the simulation are shown. These three sections are shown in Figure 56.

The windows in these sections may be resized, moved, and reordered as needed.

The simulation window also includes a menu bar and toolbars for controlling the simulation and the waveform display.

Title bar The title bar of the simulation window (the area at the top of the window) identifies the name of the currently open simulation (either simulation profile or circuit file) and the name of the currently active document displayed in the main window area. For example, the simulation window shown in Figure 56 indicates that simulation profile Example-TRAN is currently open and the active document displayed is Example-Example-TRAN.DAT.

Menus and Toolbars The menus accessed from the menu bar include commands to set up and control the simulator, customize the window display characteristics, and configure the way the waveforms are displayed. The toolbar buttons duplicate many of the more frequently used commands.

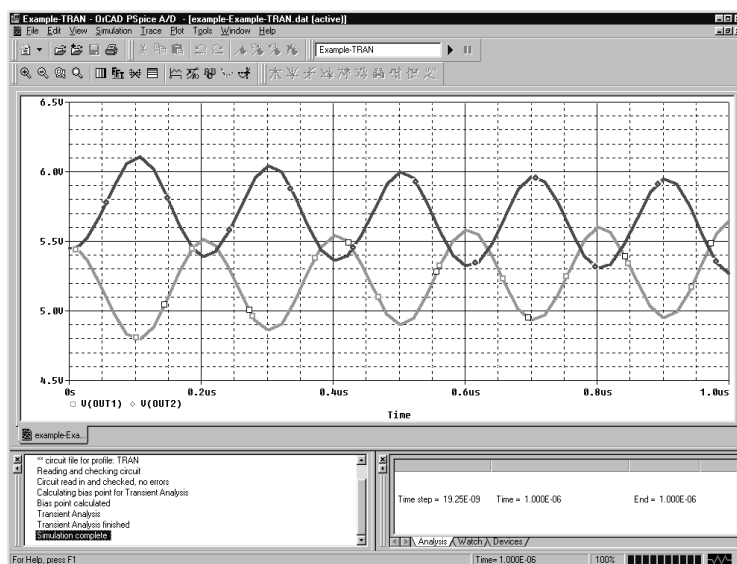


Figure 55 *PSpice simulation window*

Main window section The top central portion (by default) of the simulation window is the main window section where documents (such as waveforms, circuit description, output information etc.) are displayed within child windows. These windows are tabbed by default. The tabs at the bottom left show the names of the documents that each child window contains. Clicking on a tab brings that child window to the foreground. Figure 56 shows the tabbed document windows for Example-Example-TRAN.DAT and Example-Example-TRAN.OUT.

You can configure the display of these windows to suit your preferences and to make the analysis of the circuit quick and readily understandable. These windows can also be resized, moved, and reordered to suit your needs.

Output window section The lower left portion of the simulation window provides a listing of the output from the simulation. It shows informational, warning, and error messages from the simulation. You can resize and relocate this window to make it easier to read.

Simulation status window section The lower right portion of the simulation window presents a set of tabbed windows that show detailed status about the simulation. There are three tabbed windows in this section: the Analysis window, the Watch Variable window, and the Devices window. The Analysis window provides a running log of values of simulation variables (parameters such as Temperature, Time Step, and Time). The Watch Variable window displays watch variables and their values. These are the variables setup to be monitored during simulation. The Devices window displays the devices that are being simulated.

DC analyses

8

Chapter overview

This chapter describes how to set up DC analyses and includes the following sections:

- [DC Sweep on page 8-214](#)
- [Bias point on page 8-223](#)
- [Small-signal DC transfer on page 8-225](#)
- [DC sensitivity on page 8-228](#)

DC Sweep

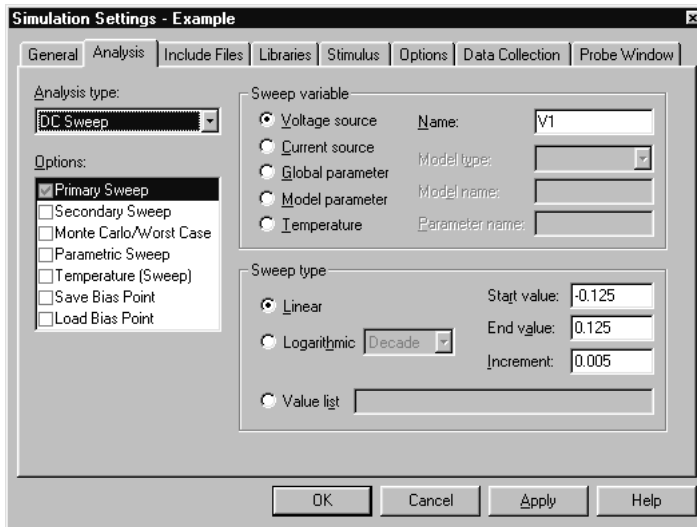
Minimum requirements to run a DC sweep analysis

Minimum circuit design requirements

Table 9 *DC sweep circuit design requirements*

Swept variable type	Requirement
voltage source	voltage source with a DC specification (VDC, for example)
temperature	none
current source	current source with a DC specification (IDC, for example)
model parameter	PSpice model (.MODEL)
global parameter	global parameter defined with a parameter block (.PARAM)

Minimum program setup requirements



- 1 In Capture, select New Simulation Profile or Edit Simulation Settings from the PSpice menu. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.

- 2 Under Analysis type, select DC Sweep.
- 3 For the Primary Sweep option, enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.
- 4 Click OK to save the simulation profile.
- 5 Select Run under the PSpice menu to start the simulation.

Note Do not specify a DC sweep and a parametric analysis for the same variable.

Overview of DC sweep

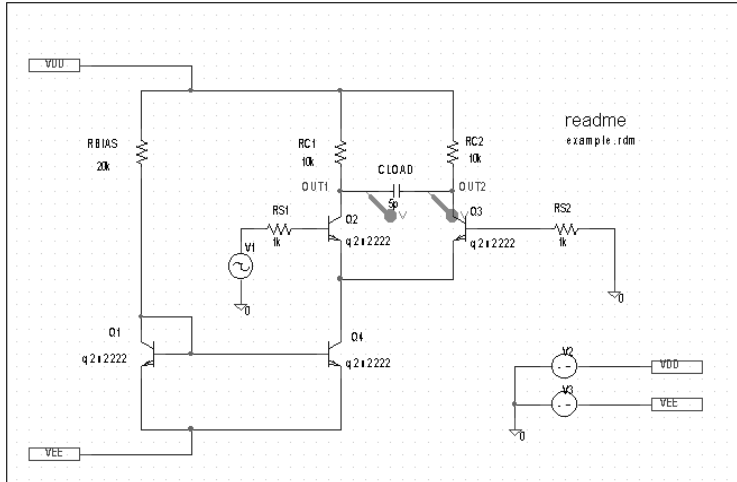
The DC sweep analysis causes a DC sweep to be performed on the circuit. DC sweep allows you to sweep a source (voltage or current), a global parameter, a model parameter, or the temperature through a range of values. The bias point of the circuit is calculated for each value of the sweep. This is useful for finding the transfer function of an amplifier, the high and low thresholds of a logic gate, and so on.

For the DC sweep analysis specified in Figure 56, the voltage source V1 is swept from -0.125 volts to 0.125 volts by steps of 0.005. This means that the output has $(0.125 - (-0.125))/0.005 + 1 = 51$ steps or simulation points.

A source with a DC specification (such as VDC or IDC) must be used if the swept variable is to be a voltage type or current source. To set the DC value, select Properties from the Edit menu, then click on the cell under the DC column and type in its value.

The default DC value of V1 is overridden during the DC sweep analysis and is made to be the swept value. All of the other sources retain their values.

After running the analysis, the simulation output file (EXAMPLE.OUT for the EXAMPLE.OPJ circuit in Figure 56) contains a table of voltages relating V1, node OUT1, and node OUT2.



The example circuit EXAMPLE.OPJ is provided with the OrCAD program installation.

Figure 56 Example schematic EXAMPLE.OPJ.

To calculate the DC response of an analog circuit, PSpice removes time from the circuit. This is done by treating all capacitors as open circuits, all inductors as shorts, and using only the DC values of voltage and current sources.

In order to solve the circuit equations, PSpice uses an iterative algorithm. For analog devices, the equations are continuous.

Setting up a DC stimulus

To run a DC sweep or small-signal DC transfer analysis, you need to place and connect one or more independent sources and then set the DC voltage or current level for each source.

To set up a DC stimulus

- 1 Place and connect one of these symbols in your schematic:

Table 10

For voltage input	
Use this...	When you are running...
VDC	A DC Sweep and/or Bias Point (transfer function) analysis only.
VSRC	Multiple analysis types including DC Sweep and/or Bias Point (transfer function).

Table 11

For current input	
Use this...	When you are running...
IDC	A DC Sweep and/or Bias Point (transfer function) analysis only.
ISRC	Multiple analysis types including DC Sweep and/or Bias Point (transfer function).

- 2 Double-click the symbol instance to display the Parts spreadsheet appears.
- 3 Click in the cell under the DC column to edit its value.
- 4 Define the DC specification as follows:

If you are planning to run an AC or transient analysis in addition to a DC analysis, see the following:

- [Using time-based stimulus parts with AC and DC properties on page 3-77](#) for other source symbols that you can use.
- [Using VSRC or ISRC parts on page 3-78](#) to find out how to specify the TRAN attribute for a time-based input signal when using VSRC or ISRC symbols.

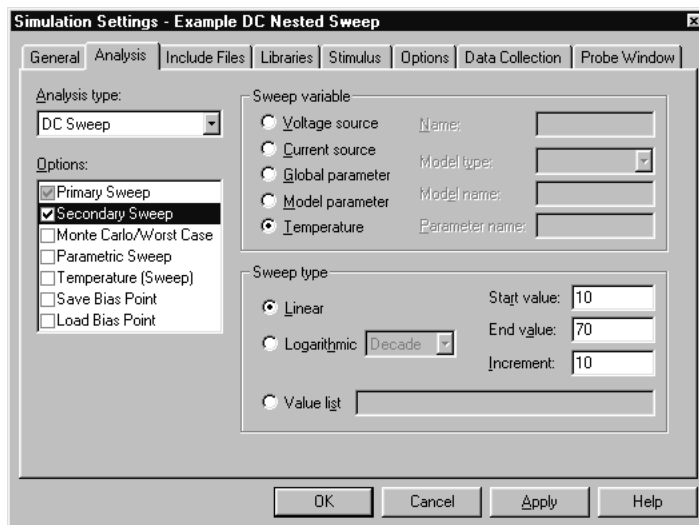
Table 12

Set this attribute...	To this value...
DC	<i>DC_level</i> where <i>DC_level</i> is in volts or amps (units are optional).

- Click OK twice to exit the dialog boxes.

Nested DC sweeps

A second sweep variable can be selected after a primary sweep value has been specified in the DC Sweep dialog box. When you specify a secondary sweep variable, it forms the outer loop for the analysis. That is, for every increment of the second sweep variable, the first sweep variable is stepped through its entire range of values.



To set up a nested sweep

- Under Options, select the Secondary Sweep box for the DC Sweep Analysis type.

- 2 Enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.

Curve families for DC sweeps

When a nested DC sweep is performed, the entire curve family is displayed. That is, the nested DC sweep is treated as a single data section (or you can think of it as a single PSpice run).

For the circuit shown in Figure 57, you can set up a DC sweep analysis with an outer sweep of the voltage source VD and an inner sweep of the voltage source VG as listed in [Table 1](#).

Table 1 *Curve family example setup*

	Outer sweep	Nested sweep
Swept Var Type	voltage source	voltage source
Sweep Type	linear	linear
Name	VD	VG
Start Value	0	0
End Value	5	2
Increment	0.1	0.5

When the DC sweep analysis is run, add a current marker at the drain pin of M1 and display the simulation results in PSpice. The result will look like Figure 58.

To add a load line for a resistor, add a trace that computes the load line from the sweep voltage. Assume that the X axis variable is the sweep voltage V_VD, which runs from 0 to 5 volts. The expression which will add a trace that is the load line for a 50 kohm resistor is:

$$(5V - V_VD) / 50K$$

This can be useful for determining the bias point for each member of a curve family as shown in Figure 59.

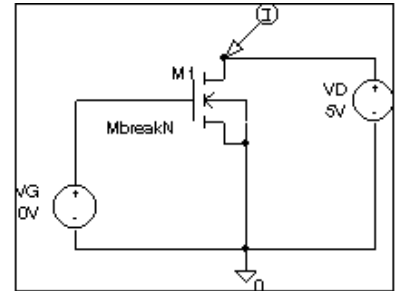
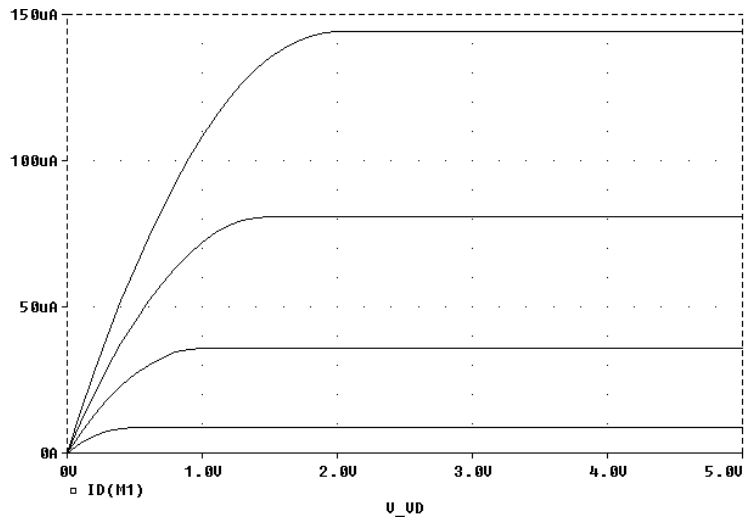
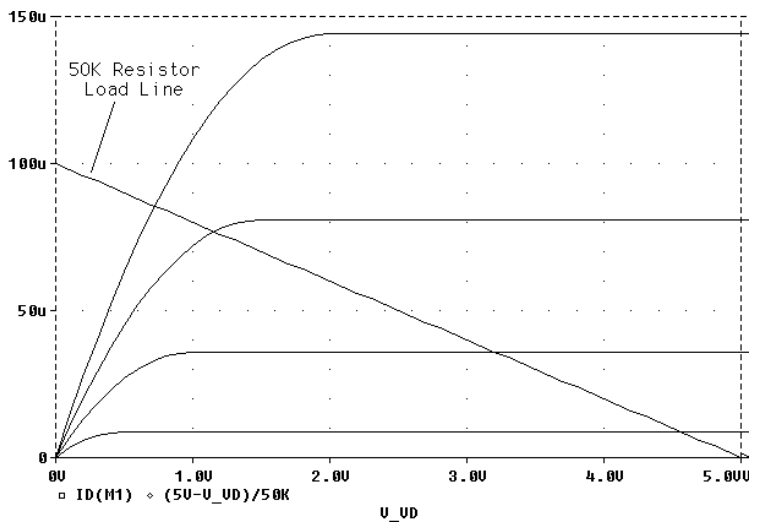


Figure 57 *Curve family example schematic.*

In Capture, from the PSpice menu, point to Markers, then choose Mark Current Into Pin to add a current marker.

V_VD is the hierarchical name for VD created by netlisting the schematic.

Figure 58 *Device curve family.*Figure 59 *Operating point determination for each member of the curve family.*

Bias point

Minimum requirements to run a bias point analysis

Minimum circuit design requirements

None.

Minimum program setup requirements

- 1 Under Analysis type in the Simulation Settings dialog box, select Bias Point.
- 2 For the General Settings option, enter the necessary parameter values and select the appropriate check boxes to complete the analysis specifications.
- 3 Click OK to save the simulation profile.
- 4 In Capture, from the PSpice menu, select Run to start the simulation.

Overview of bias point

The bias point is calculated for any analysis whether or not the Bias Point analysis is enabled in the Simulation Settings dialog box. However, additional information is reported when the Bias Point analysis is enabled.

When Bias Point analysis is not enabled, only analog node voltages and states are reported to the output file.

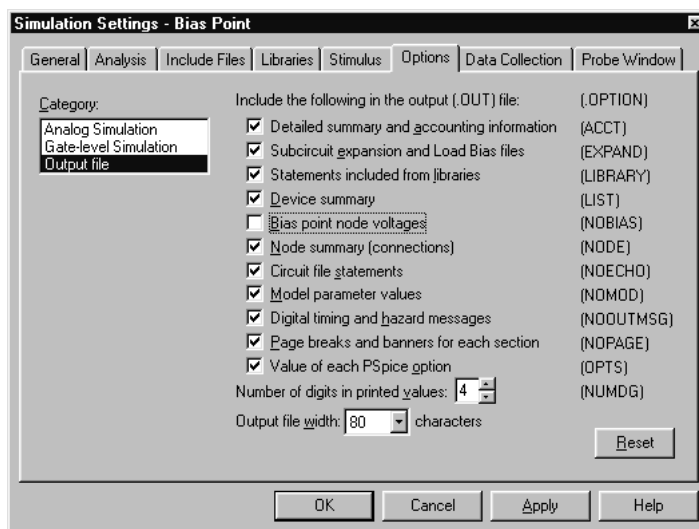
Also see [Save and load bias point on page A-374](#).

When the Bias Point analysis is enabled, the following information is reported to the output file:

- a list of all analog node voltages
- the currents of all voltage sources and their total power
- a list of the small-signal parameters for all devices

If Bias Point is enabled, you can suppress the reporting of the bias point analog node values, as follows:

- 1 Under the Options tab in the Simulation Settings dialog box, select Output file in the Category box.
- 2 Uncheck the box for Bias point node voltages (NOBIAS).



Small-signal DC transfer

Minimum requirements to run a small-signal DC transfer analysis

Minimum circuit design requirements

- The circuit should contain an input source, such as VSRC.

Minimum program setup requirements

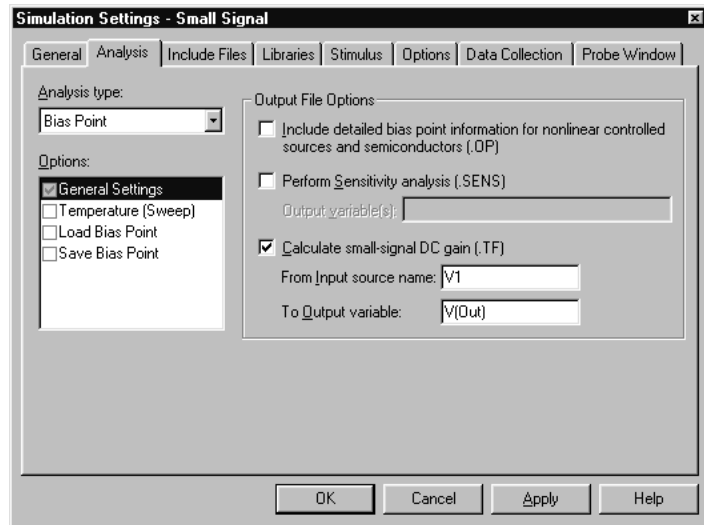
- 1 Under Analysis type in the Simulation Settings dialog box, select Bias Point.
- 2 Specify the name of the input source desired. See [Output variables on page 7-199](#) for a description of output variable formats.
- 3 Click OK to save the simulation profile.
- 4 In Capture, from the PSpice menu, select Run to start the simulation.

Overview of small-signal DC transfer

The small-signal DC transfer analysis calculates the small-signal transfer function by transforming the circuit around the bias point and treating it as a linear circuit. The small-signal gain, input resistance, and output resistance are calculated and reported.

For N and O devices in the analog interface subcircuits, has a well-defined linear equivalent.

To calculate the small-signal gain, input resistance, and output resistance



- 1 In the Bias Point dialog box, select Calculate small-signal DC gain (.TF).
- 2 Specify the value for either an output voltage or the current through a voltage source in the To Output variable box.

For example, entering V(a,b) as the output variable specifies that the output variable is the output voltage between two nets, a and b. Entering I(VDRIV) as the output variable specifies that the output variable is the current through a voltage source VDRIV.

- 3 Specify the input source name in the Calculate small-signal DC gain (.TF) portion of the Bias Point dialog box.

The gain from the input source to the output variable is calculated along with the input and output resistances.

For example, if you enter `V(OUT2)` as the output variable and `V1` as the input source, the input resistance for `V1` is calculated, the output resistance for `V(OUT2)` is calculated, and the gain from `V1` to `V(OUT2)` is calculated. All calculations are reported to the simulation output file.

DC sensitivity

Minimum requirements to run a DC sensitivity analysis

Minimum circuit design requirements

None.

Minimum program setup requirements

- 1 In the Bias Point dialog box, select Perform Sensitivity analysis (.SENS).
- 2 Enter the required value(s) in the Output variable(s) box.
- 3 Click OK to save the simulation profile. (Be sure you give the new profile an appropriate name under the General tab prior to saving.)
- 4 In Capture, from the PSpice menu, select Run to start the simulation.

Overview of DC sensitivity

DC sensitivity analysis calculates and reports the sensitivity of one node voltage to each device parameter for the following device types:

- resistors
- independent voltage and current sources
- voltage and current-controlled switches
- diodes
- bipolar transistors

The sensitivity is calculated by linearizing all devices around the bias point.

AC analyses

9

Chapter overview

This chapter describes how to set up AC sweep and noise analyses.

- [AC sweep analysis on page 9-232](#) describes how to set up an analysis to calculate the frequency response of your circuit. This section also discusses how to define an AC stimulus and how PSpice treats nonlinear devices in an AC sweep.
- [Noise analysis on page 9-241](#) describes how to set up an analysis to calculate device noise contributions and total input and output noise.

AC sweep analysis

Setting up and running an AC sweep

The following procedure describes the minimum setup requirements for running an AC sweep analysis. For more detail on any step, go to the pages referenced in the sidebars.

To set up and run an AC sweep

To find out how, see [Setting up an AC stimulus on page 9-233](#).

- 1 Place and connect a voltage or current source with an AC input signal.
- 2 From the PSpice menu, select New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.

To find out how, see [Setting up an AC analysis on page 9-235](#).

- 3 Choose AC Sweep/Noise in the Analysis type list box.
- 4 Specify the required parameters for the AC sweep or noise analysis you want to run.
- 5 Click OK to save the simulation profile.
- 6 From the PSpice menu, select Run to start the simulation.

What is AC sweep?

AC sweep is a frequency response analysis. PSpice calculates the small-signal response of the circuit to a combination of inputs by transforming it around the bias point and treating it as a linear circuit. Here are a few things to note:

To find out more, see [How PSpice treats nonlinear devices on page 9-239](#).

- Nonlinear devices, such as voltage- or current-controlled switches, are transformed to linear

circuits about their bias point value before PSpice runs the linear (small-signal) analysis.

- Because AC sweep analysis is a linear analysis, it only considers the gain and phase response of the circuit; it does not limit voltages or currents.

The best way to use AC sweep analysis is to set the source magnitude to one. This way, the measured output equals the gain, relative to the input source, at that output.

Setting up an AC stimulus

To run an AC sweep analysis, you need to place and connect one or more independent sources and then set the AC magnitude and phase for each source.

To set up an AC stimulus

- 1 Place and connect one of these symbols in your schematic:

Table 2

For voltage input	
Use this...	When you are running...
VAC	An AC sweep analysis only.
VSRC	Multiple analysis types including AC sweep.

Table 3

For current input	
Use this...	When you are running...
IAC	An AC sweep analysis only.
ISRC	Multiple analysis types including AC sweep.

- 2 Double-click the symbol instance to display the Parts spreadsheet.

Note Unlike DC sweep, the AC Sweep/Noise dialog box not include an input source option. Instead, each independent source in your circuit contains its own AC specification for magnitude and phase.

If you are planning to run a DC or transient analysis in addition to an AC analysis, see [If you want to specify multiple stimulus types on page 3-77](#) for additional information and source symbols that you can use.

- 3 Click in the cell under the appropriate property column to edit its value. Depending on the source symbol that you placed, define the AC specification as follows:

Table 4

For VAC or IAC	
Set this property...	To this value...
ACMAG	AC magnitude in volts (for VAC) or amps (for IAC); units are optional.
ACPHASE	Optional AC phase in degrees.

Table 5

For VSRC or ISRC	
Set this property...	To this value...
AC	<i>Magnitude_value</i> [<i>phase_value</i>] where <i>magnitude_value</i> is in volts or amps (units are optional) and the optional <i>phase_value</i> is in degrees.

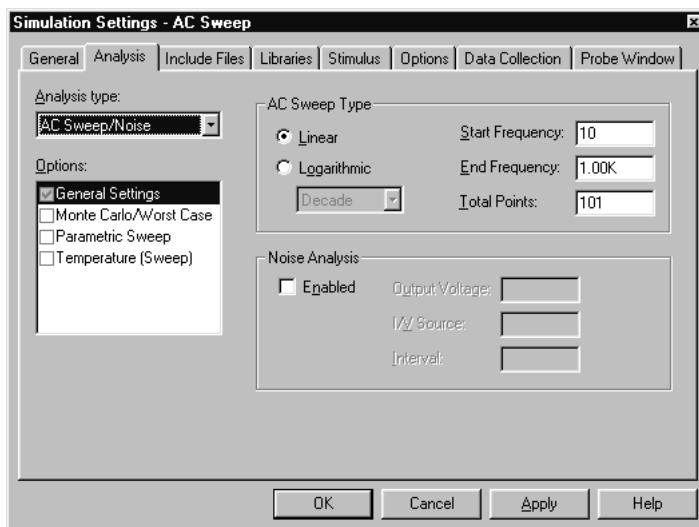
If you are also planning to run a transient analysis, see [Using VSRC or ISRC parts on page 3-78](#) to find out how to specify the TRAN property.

Setting up an AC analysis

To set up the AC analysis

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.



- 2 Choose AC Sweep/Noise in the Analysis type list box.
- 3 Under Options, select General Settings if it is not already enabled.
- 4 Set the number of sweep points as follows:

Table 6

To sweep frequency...	Do this...
linearly	Under AC Sweep Type, click Linear, and enter the total number of points in the sweep in the Total Points box.
logarithmically by decades	Under AC Sweep Type, click Logarithmic, select Decade (default), and enter the total number of points per decade in the Total Points box.
logarithmically by octaves	Under AC Sweep Type, click Logarithmic, select Octave, and enter the total number of points per octave in the Total Points box.

If you also want to run a noise analysis, then before clicking OK, complete the Noise Analysis frame in this dialog box as described in [Setting up a noise analysis on page 9-243](#).

- 5 In the Start Frequency and End Frequency text boxes, enter the starting and ending frequencies, respectively, for the sweep.
- 6 Click OK to save the simulation profile.

AC sweep setup in example.opj

If you look at the example circuit, EXAMPLE.OPJ, provided with your OrCAD programs, you'll find that its AC analysis is set up as shown in Figure 61.

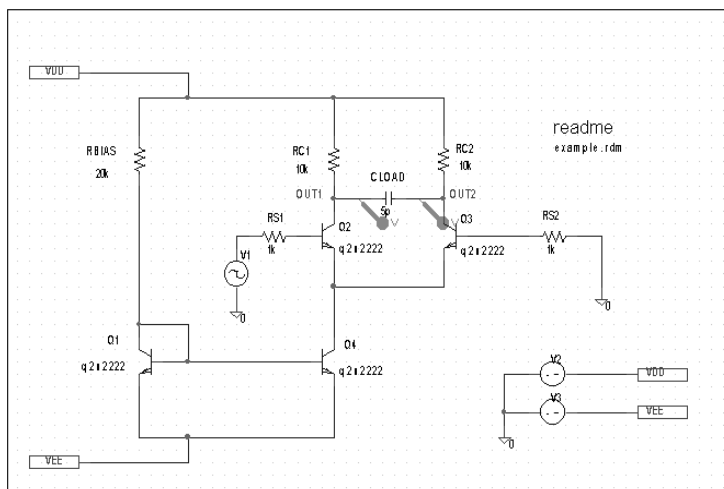


Figure 60 Circuit diagram for EXAMPLE.OPJ.

Frequency is swept from 100 kHz to 10 GHz by decades, with 10 points per decade. The V1 independent voltage source is the only input to an amplifier, so it is the only AC stimulus to this circuit. Magnitude equals 1 V and relative phase is left at zero degrees (the default). All other voltage sources have zero AC value.

Note The source, V1, is a VSIN source that is normally used for setting up sine wave signals for a transient analysis. It also has an AC property so that you can use it for an AC analysis.

To find out more about VSIN and other source symbols that you can use for AC analysis, see [Using time-based stimulus parts with AC and DC properties on page 3-77](#).

Note The source, V1, is a VSIN source that is normally used for setting up sine wave signals for a transient analysis. It also has an AC property so that you can use it for an AC analysis.

To find out more about VSIN and other source symbols that you can use for AC analysis, see [Using time-based stimulus parts with AC and DC properties on page 3-77](#).

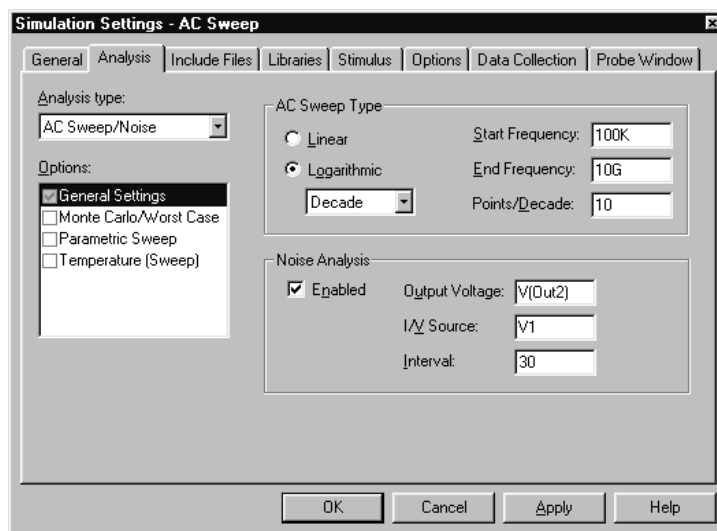


Figure 61 AC analysis setup for EXAMPLE.OPJ.

Frequency is swept from 100 kHz to 10 GHz by decades, with 10 points per decade. The V1 independent voltage source is the only input to an amplifier, so it is the only AC stimulus to this circuit. Magnitude equals 1 V and relative phase is left at zero degrees (the default). All other voltage sources have zero AC value.

How PSpice treats nonlinear devices

An AC Sweep analysis is a linear or small-signal analysis. This means that nonlinear devices must be linearized to run the analysis.

What's required to transform a device into a linear circuit

In order to transform a device (such as a transistor amplifier) into a linear circuit, you must do the following:

- 1 Compute the DC bias point for the circuit.
- 2 Compute the complex impedance and/or transconductance values for each device at this bias point.
- 3 Perform the linear circuit analysis at the frequencies of interest by using simplifying approximations.

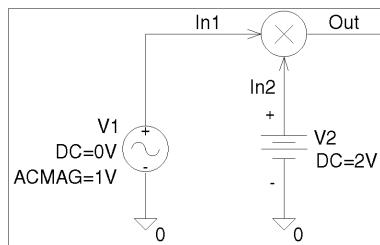
Example: Replace a bipolar transistor in common-emitter mode with a constant transconductance (collector current proportional to base-emitter voltage) and a number of constant impedances.

What PSpice does

PSpice automates this process for you. PSpice computes the partial derivatives for nonlinear devices at the bias point and uses these to perform small-signal analysis.

Example: nonlinear behavioral modeling block

Suppose you have an analog behavioral modeling block that multiplies $V(1)$ by $V(2)$. Multiplication is a nonlinear operation. To run an AC sweep analysis on this block, the block needs to be replaced with its linear equivalent. To determine the linear equivalent block, PSpice needs a known bias point.



Using a DC source

Consider the circuit shown here. At the DC bias point, PSpice calculates the partial derivatives which determine the linear response of the multiplier as follows:

$$\begin{aligned} V(Out) &= V(In1) \cdot \frac{\partial V(Out)}{\partial V(In1)} + V(In2) \cdot \frac{\partial V(Out)}{\partial V(In2)} \\ &= V(In1) \cdot V(In2) + V(In2) \cdot V(In1) \end{aligned}$$

For this circuit, this equation reduces to:

$$V(Out) = V(In1) \cdot 2 + V(In2) \cdot 0$$

This means that the multiplier acts as an amplifier of the AC input with a gain that is set by the DC input.



This is exactly how a double-balanced mixer behaves. In practice, this is a simple multiplier.

Note *A double-balanced mixer with inputs at the same frequency would produce outputs at DC at twice the input frequency, but these terms cannot be seen with a linear, small-signal analysis.*

Caution: multiplying AC sources

Suppose that you replace the 2 volt DC source in this example with an AC source with amplitude 1 and no DC value (DC=0). When PSpice computes the bias point, there are no DC sources in the circuit, so all nodes are at 0 volts at the bias point. The linear equivalent of the multiplier block is a block with gain 0, which means that there is no output voltage at the fundamental frequency.

Noise analysis

Setting up and running a noise analysis

The following procedure describes the minimum setup requirements for running a noise analysis. For more detail on any step, go to the pages referenced in the sidebars.

To set up and run an AC sweep

- 1 Place and connect a voltage or current source with an AC input signal.
- 2 Set up the AC sweep simulation specifications.
- 3 Set up the noise simulation specifications and enable the analysis in the AC Sweep/Noise portion of the Simulation Settings dialog box.
- 4 Click OK to save the simulation profile.
- 5 From the PSpice menu, choose Run to start the simulation.

To find out how, see [Setting up an AC stimulus on page 9-233](#).

To find out how, see [Setting up an AC analysis on page 9-235](#).

To find out how, see [Setting up a noise analysis on page 9-243](#).

What is noise analysis?

When running a noise analysis, PSpice calculates and reports the following for each frequency specified for the AC Sweep/Noise analysis:

- Device noise, which is the noise contribution propagated to the specified output net from every resistor and semiconductor device in the circuit; for semiconductor devices, the device noise is also broken down into constituent noise contributions where applicable
- Total output and equivalent input noise

Example: Diodes have separate noise contributions from thermal, shot, and flicker noise.

Table 7

This value...	Means this...
Output noise	RMS sum of all the device contributions propagated to a specified output net
Input noise	equivalent noise that would be needed at the input source to generate the calculated output noise in an ideal (noiseless) circuit

How PSpice calculates total output and input noise

To calculate total noise at an output net, PSpice computes the RMS sum of the noise propagated to the net by all noise-generating devices in the circuit.

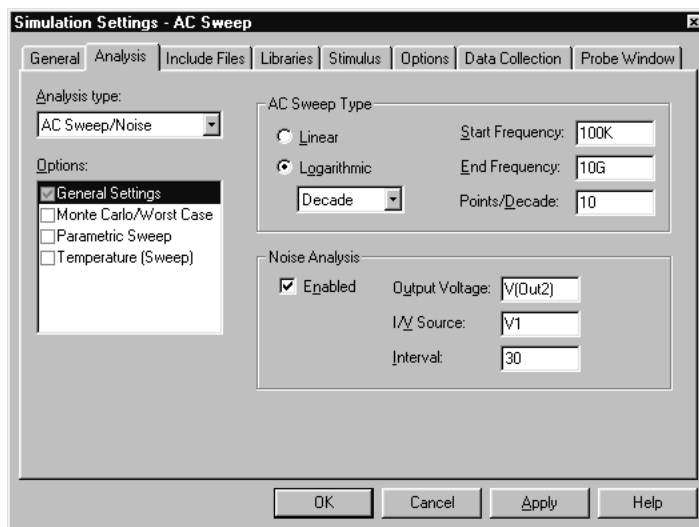
To calculate the equivalent input noise, PSpice then divides total output noise by the gain from the input source to the output net. This results in the amount of noise which, if injected at the input source into a noiseless circuit, would produce the total noise originally calculated for the output net.

Setting up a noise analysis

To set up the noise analysis

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.



- 2 Choose AC Sweep/Noise in the Analysis type list box.
- 3 Under Options, select General Settings if it is not already enabled.
- 4 Specify the AC sweep analysis parameters as described on page [9-235](#).
- 5 Enable the Noise Analysis check box.
- 6 Enter the noise analysis parameters as follows:

To find out more about valid syntax, see [Output variables on page 7-199](#).

Example: U1.V2

Note *In the Probe window, you can view the device noise contributions at every frequency specified in the AC sweep. The Interval parameter has no effect on what PSpice writes to the Probe data file.*

Table 8

In this text box...	Type this...
Output Voltage	A voltage output variable of the form $V(node, [node])$ where you want the total output noise calculated.
I/V Source	The name of an independent current or voltage source where you want the equivalent input noise calculated. Note <i>If the source is in a lower level of a hierarchical schematic, separate the names of the hierarchical devices with periods (.).</i>
Interval	An integer n designating that at every n^{th} frequency, you want to see a table printed in the PSpice output file (.out) showing the individual contributions of all of the circuit's noise generators to the total noise.

7 Click OK to save the simulation profile.

Analyzing Noise in the Probe window

You can use these output variable formats to view traces for device noise contributions and total input or output noise at every frequency in the analysis.

For a break down of noise output variables by supported device type, see [Table 6 on page 13-358](#).

To view this...	Use this output variable...	Which is represented by this equation* ...
Flicker noise for a device	NFID(<i>device_name</i>) NFIB(<i>device_name</i>)	$\text{noise} \propto k_f \cdot \frac{I_f^{a_f}}{f^b}$
Shot noise for a device	NSID(<i>device_name</i>) NSIB(<i>device_name</i>) NSIC(<i>device_name</i>)	For diodes and BJTs: $\text{noise} \propto 2qI$ For GaAsFETs, JFETs, and MOSFETs: $\text{noise} \propto 4kT \cdot \frac{dI}{dV} \cdot \frac{2}{3}$
Thermal noise for the RB, RC, RD, RE, RG, or RS constituent of a device, respectively	NRB(<i>device_name</i>) NRC(<i>device_name</i>) NRD(<i>device_name</i>) NRE(<i>device_name</i>) NRG(<i>device_name</i>) NRS(<i>device_name</i>)	$\text{noise} \propto \frac{4kT}{R}$
Total noise for a device	NTOT(<i>device_name</i>)	Sum of all contributors in <i>device_name</i>
Total output noise for the circuit	NTOT(ONoise)	$\sum_{\text{device}} \text{NTOT}(\text{device})$
RMS-summed output noise for the circuit	V(ONoise)	RMS sum of all contributors ($\sqrt{\text{NTOT}(\text{ONoise})}$)
Equivalent input noise for the circuit	V(INoise)	$\frac{V(\text{ONoise})}{\text{gain}}$

* To find out more about the equations that describe noise behavior, refer to the appropriate device type in the *Analog Devices* chapter in the *OrCAD PSpice Reference Manual*.

About noise units

Table 9

This type of noise output variable...	Is reported in these units...
Device contribution of the form Nxxx	$(volts)^2/(Hz)$
Total input or output noise of the form V(ONoise) or V(INoise)	$(volts)/(\sqrt{Hz})$

Example

You can run a noise analysis on the circuit shown in Figure 60 on page 9-237.

To run a noise analysis on the example:

In Capture, open the EXAMPLE.DSN circuit provided with your OrCAD programs in the ORCAD\CAPTURE\SAMPLES subdirectory.

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.
- 2 Choose AC Sweep/Noise in the Analysis type list box.
- 3 Under Options, select General Settings if it is not already enabled.
- 4 Enable the Noise Analysis check box.
- 5 Enter the following parameters for the noise analysis:

Output Voltage	V(OUT2)
I/V Source	V1
Interval	30

For a description of the Interval parameter, see page [9-244](#).

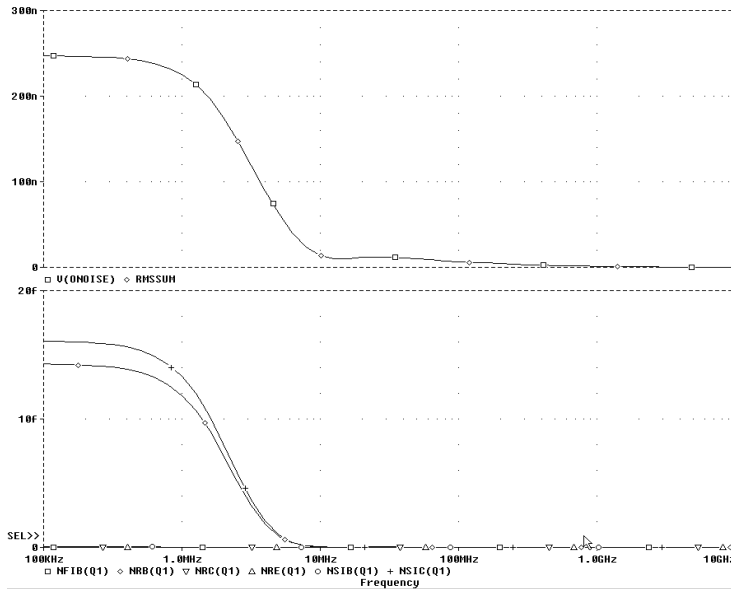
These settings mean that PSpice will calculate noise contributions and total output noise at net OUT2 and equivalent input noise from V1.

Figure 63 shows Probe traces for Q1's constituent noise sources as well as total noise for the circuit after simulating. Notice that the trace for RMSSUM (at the top of the plot), which is a macro for the trace expression

$$\text{SQRT}(\text{NTOT}(\text{Q1}) + \text{NTOT}(\text{Q2}) + \text{NTOT}(\text{Q3}) + \dots),$$

exactly matches the total output noise, V(ONoise), calculated by PSpice.

To find out more about PSpice macros, refer to PSpice A/D online Help.



Note The source, V1, is a VSIN source that is normally used for setting up sine wave signals for a transient analysis. It also has an AC property so that you can use it for an AC analysis.

To find out more about VSIN and other source symbols that you can use for AC analysis, see [Using time-based stimulus parts with AC and DC properties on page 3-77](#).

Figure 62 Device and total noise traces for EXAMPLE.DSN.

Frequency is swept from 100 kHz to 10 GHz by decades, with 10 points per decade. The V1 independent voltage source is the only input to an amplifier, so it is the only AC stimulus to this circuit. Magnitude equals 1 V and relative phase is left at zero degrees (the default). All other voltage sources have zero AC value.

Transient analysis

10

Chapter overview

This chapter describes how to set up a transient analysis and includes the following sections:

- [Overview of transient analysis on page 10-250](#)
- [Defining a time-based stimulus on page 10-252](#)
- [Transient \(time\) response on page 10-263](#)
- [Internal time steps in transient analyses on page 10-265](#)
- [Switching circuits in transient analyses on page 10-266](#)
- [Plotting hysteresis curves on page 10-266](#)
- [Fourier components on page 10-268](#)

Overview of transient analysis

Minimum requirements to run a transient analysis

Minimum circuit design requirements

Circuit should contain one of the following:

- An independent source with a transient specification (see [Table 10](#))
- An initial condition on a reactive element
- A controlled source that is a function of time

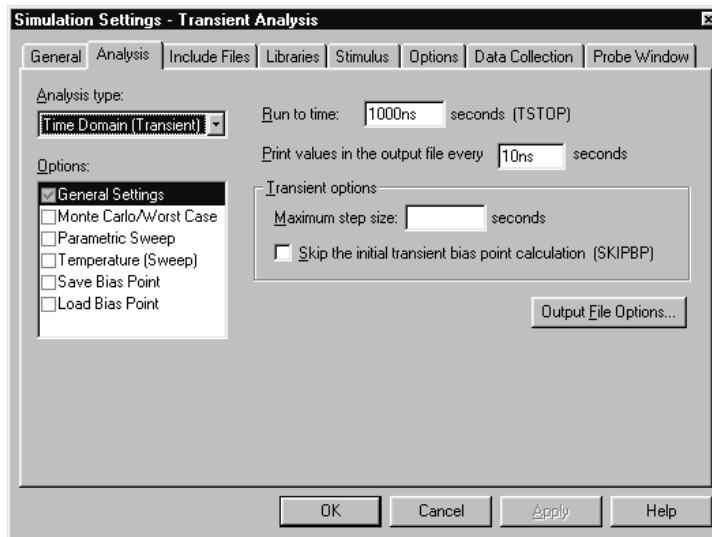
Minimum program setup requirements

See [Setting up analyses on page 7-197](#) for a description of the Analysis Setup dialog box.

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.

- 2 From the Analysis type list box, select Time Domain (Transient).
- 3 Specify the required parameters for the transient analysis you want to run.
- 4 Click OK to save the simulation profile.
- 5 From the PSpice menu, choose Run to start the simulation.



Defining a time-based stimulus

Overview of stimulus generation

Symbols that generate input signals for your circuit can be divided into two categories:

- those whose transient behavior is characterized graphically using the Stimulus Editor
- those whose transient behavior is characterized by manually defining their properties within Capture

Their symbols are summarized in [Table 10](#).

Table 10 *Stimulus symbols for time-based input signals*

Specified by...	Symbol name	Description
Using the Stimulus Editor	VSTIM	voltage source
	ISTIM	current source
Defining symbol attribute	VSRC	voltage sources
	VEXP	
	VPULSE	
	VPWL	
	VPWL_RE_FOREVER	
	VPWL_F_RE_FOREVER	
	VPWL_N_TIMES	
	VPWL_F_N_TIMES	
	VSFFM	
	VSIN	
	ISRC	current sources
	IEXP	
	IPULSE	
	IPWL	
	IPWL_RE_FOREVER	
	IPWL_F_RE_FOREVER	
	IPWL_N_TIMES	
	IPWL_F_N_TIMES	
	ISFFM	
	ISIN	

To use any of these source types, you must place the symbol in your schematic and then define its transient behavior.

Each property-characterized stimulus has a distinct set of attributes depending upon the kind of transient behavior it represents. For VPWL_F_xxx, IPWL_F_xxx, and FSTIM, a separate file contains the stimulus specification.

As an alternative, the Stimulus Editor utility automates the process of defining the transient behavior of stimulus devices. The Stimulus Editor allows you to create analog stimuli which generate sine wave, repeating pulse, exponential pulse, single-frequency FM, and piecewise linear waveforms.

The stimulus specification created using the Stimulus Editor is saved to a file, automatically configured into the schematic, and associated with the corresponding VSTIM or ISTIM part instance or symbol definition.

The Stimulus Editor utility

The Stimulus Editor is a utility that allows you to quickly set up and verify the input waveforms for a transient analysis. You can create and edit voltage sources, and current sources for your circuit. Menu prompts guide you to provide the necessary parameters, such as the rise time, fall time, and period of an analog repeating pulse. Graphical feedback allows you quickly verify the waveform.

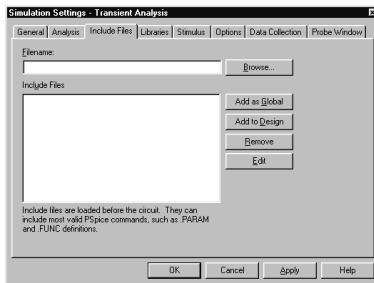
OrCAD program versions without the Stimulus Editor must use the characterized-by-property sources listed in [Table 10 on page 10-252](#).

Stimulus files

The Stimulus Editor produces a file containing the stimuli with their transient specification. These stimuli are defined as simulator device declarations using the V (voltage source) and I (current source) forms. Since the

Stimulus Editor produces these statements automatically, you will never have to be concerned with their syntax. However, if you are interested in a detailed description of their syntax, see the descriptions of V and I devices in the *Analog Devices* chapter of the the online *OrCAD PSpice A/D Reference Manual*.

Configuring stimulus files



The Include Files tab in the Simulation Settings dialog box allows you to view the list of stimulus files pertaining to your current schematic. You can also manually add, delete, or change the stimulus file configuration in this tab dialog box. The list box displays all of the currently configured stimulus files. One file is specified per line. Files can be configured as either global to the Capture environment or local to the current design. Global files are marked with an asterisk (*) after the file name.

When starting the Stimulus Editor from Capture, stimulus files are automatically configured (added to the list) as local to the current design. Otherwise, new stimulus files can be added to the list by entering the file name in the Filename text box and then clicking the Add to design (local configuration) or Add as global (global configuration) button.

Starting the Stimulus Editor

The Stimulus Editor is fully integrated with Capture and can be run from either the schematic editor or symbol editor.

You can start the Stimulus Editor by doing the following:

- 6 Select one or more stimulus instances in the schematic
- 7 From the Edit menu, choose PSpice Stimulus.

When you first start the Stimulus Editor, you may need to adjust the scale settings to fit the trace you are going to add. You can use Axis Settings on the Plot menu or the corresponding toolbar button to change the displayed data, the extent of the scrolling region, and the minimum resolution for each of the axes. Displayed Data Range parameters determine what portion of the stimulus data set will be presented on the screen. Extent of Scrolling Region parameters set the absolute limits on the viewable range. Minimum Resolution parameters determine the smallest usable increment (example: if it is set to 1 msec, then you cannot add a data point at 1.5 msec).

Defining stimuli

- 1 Place stimulus part instances from the symbol set: VSTIM, ISTIM and DIGSTIMn.
- 2 Click the source instance to select it.
- 3 From the Edit menu, choose PSpice Stimulus to start the Stimulus Editor.
- 4 Fill in the transient specification according to the dialogs and prompts.
- 5 From the File menu, choose Save to save the edits.

Example: piecewise linear stimulus

- 1 Open an existing schematic or start a new one.
- 2 From the Place menu, choose Part and browse the SOURCE.OLB part library file for VSTIM (and select it).
- 3 Place the part. It looks like a regular voltage source with an implementation property displayed.
- 4 Click the implementation label and type `vfirst`. This names the stimulus that you are going to create.
- 5 If you are working in a new schematic, use Save from the File menu to save it. This is necessary since the schematic name is used to create the default stimulus file name.
- 6 Click the VSTIM part to select it.
- 7 From the Edit menu, choose PSpice Stimulus. This starts the Stimulus Editor and displays the New Stimulus dialog box. You can see that the stimulus already has the name of Vfirst.
- 8 Select PWL in the dialog box and click OK. The cursor looks like a pencil. The message in the status bar at the bottom of the screen lets you know that you are in the process of adding new data points to the stimulus. The left end of the bottom status bar displays the current coordinates of the cursor.

- 9 Move the cursor to (200ns, 1) and click the left mouse button. This adds the point. Notice that there is automatically a point at (0,0). Ignore it for now and continue to add a couple more points to the right of the current one.
- 10 Click-right to stop adding points.
- 11 From the File menu, choose Save.

If you make a mistake or want to make any changes, reshape the trace by dragging any of the handles to a new location. The dragged handle cannot pass any other defined data point.

To delete a point, click its handle and press **[Del]**.

To add additional points, either choose Add Point from the Edit menu, press **[Alt]+[A]**, or click the Add Point toolbar button.

At this point you can return to Capture, edit the current stimulus, or go on to create another.

Example: sine wave sweep

- 1 Open an existing schematic or start a new one.
- 2 Place a VSTIM part on your schematic.
- 3 To name the stimulus, double-click the implementation property and type `Vsin`.
- 4 Click the VSTIM part to select it.
- 5 From the PSpice menu, choose Edit Stimulus to start the Stimulus Editor.
- 6 Define the stimulus parameter for amplitude:
 - a From the New Stimulus dialog box, choose Cancel.
 - b From the Tools menu, choose Parameters.
 - c Enter `AMP=1` in the Definition text box, and click OK.
 - d From the Stimulus menu, choose New or click the New Stimulus button in the toolbar.

This example creates a 10 k sine wave with the amplitude parameterized so that it can be swept during a simulation.

- e Give the stimulus the name of Vsin.
 - f Select SIN as the type of stimulus to be created, and click OK.
- 7 Define the other stimulus properties:
- a Enter 0 for Offset Value.
 - b Enter {AMP} for Amplitude. The curly braces are required. They indicate that the expression needs to be evaluated at simulation time.
 - c Enter 10k for Frequency and click OK.
 - d From the File menu, choose Save.
- 8 Within Capture, place and define the PARAM symbol:
- a From the Place menu, choose Part.
 - b Either browse SPECIAL.OLB for the PARAM part or type in the name.
 - c Place the part on your schematic and double-click it.
 - d Click New to add a new user property.
 - e Set the value property name to AMP (no curly braces).
 - f Set the value of the VALUE1 property to 1.
- 9 Set up the parametric sweep and other analyses:
- a From the PSpice menu, choose Stimulus Editor, and click the Parametric Sweep button.
 - b Select Global Parameter in the Swept Var. Type frame.
 - c Select Linear in the Sweep type frame.
- 10 Enter AMP in the Name text box.
- 11 Specify values for the Start Value, End Value, and Increment text boxes.

You can now set up your usual Transient, AC, or DC analysis and run the simulation.

Creating new stimulus symbols

- 1 Use the Capture part editor to edit or create a part with the following properties:

Implementation Type	PSpice Stimulus
Implementation	name of the stimulus model
STIMTYPE	type of stimulus; valid values are ANALOG; if this property is nonexistent, the stimulus is assumed to be ANALOG

Editing a stimulus

To edit an existing stimulus

- 1 Start the Stimulus Editor and select Get from the Stimulus menu.
- 2 Double-click the trace name (at the bottom of the X axis for analog). This opens the Stimulus Attributes dialog box where you can modify the attributes of the stimulus directly and immediately see the effect of the changes.

PWL stimuli are a little different since they are a series of time/value pairs.

To edit a PWL stimulus

- 1 Double click the trace name. This displays the handles for each defined data point.
- 2 Click any handle to select it. To reshape the trace, drag it to a new location. To delete the data point, press **Del**.
- 3 To add additional data points, either select Add from the Edit menu or click the Add Point button.
- 4 Right-click to end adding new points.

This provides a fast way to scale a PWL stimulus.

To select a time and value scale factor for PWL stimuli

- 1 Select the PWL trace by clicking on its name.
- 2 Select Attributes from the Edit menu or click the corresponding toolbar button.

Deleting and removing traces

To delete a trace from the displayed screen, select the trace name by clicking on its name, then press **[Del]**. This will only erase the display of the trace, not delete it from your file. The trace is still available by selecting Get from the Stimulus menu.

To remove a trace from a file, select Remove from the Stimulus menu.

Note *Once a trace is removed, it is no longer retrievable. Delete traces with caution.*

Manual stimulus configuration

Stimuli can be characterized by manually starting the Stimulus Editor and saving their specifications to a file. These stimulus specifications can then be associated to stimulus instances in your schematic or to stimulus symbols in the symbol library.

To manually configure a stimulus

- 1 Start the Stimulus Editor by double-clicking on the Stimulus Editor icon in the OrCAD program group.
- 2 Open a stimulus file by selecting Open from the File menu. If the file is not found in your current library search path, you are prompted for a new file name.
- 3 Create one or more stimuli to be used in your schematic. For each stimulus:
 - a Name it whatever you want. This name will be used to associate the stimulus specification to the stimulus instance in your schematic, or to the symbol in the symbol library.
 - b Provide the transient specification.
 - c From the File menu, choose Save.

- 4 In the schematic page editor, configure the Stimulus Editor's output file into your schematic:
 - a From the Pspice menu, choose Edit Simulation Settings.
 - a In the Simulation Settings dialog box, select the Include Files tab.
 - b Enter the file name specified in step 2.
 - c If the stimulus specifications are for local use in the current design, click the Add to design button. For global use by any design, use Add as global instead.
 - d Click OK.
- 5 Modify either the stimulus instances in the schematic or symbols in the symbol library to reference the new stimulus specification.
- 6 Associate the transient stimulus specification to a stimulus instance:
 - a Place a stimulus part in your schematic from the part set: VSTIM, ISTIM, and DIGSTIMn.
 - b Click the VSTIM, ISTIM, or DIGSTIMn instance.
 - c From the Edit menu, choose Properties.
 - d Click the Implementation cell, type in the name of the stimulus, and click Apply.
 - e Complete specification of any VSTIM or ISTIM instances by selecting Properties from the Edit menu and editing their DC and AC attributes.

Click the DC cell and type its value.

Click the AC cell, type its value, and then click Apply.
 - f Close the property editor spreadsheet.
- 7 To change stimulus references globally for a part:
 - a Select the part you want to edit.
 - a From the Edit menu, choose Part to start the part editor.

- b Create or change the part definition, making sure to define the following properties:

Implementation stimulus name as defined in the Stimulus Editor

See [Chapter 5, Creating parts for models](#), for a description of how to create and edit parts.

Transient (time) response

The Transient response analysis causes the response of the circuit to be calculated from TIME = 0 to a specified time. A transient analysis specification is shown for the circuit EXAMPLE.OPJ in Figure 63. (EXAMPLE.OPJ is shown in Figure 64.)

The analysis is to span the time interval from 0 to 1000 nanoseconds and values should be reported to the simulation output file every 20 nanoseconds.

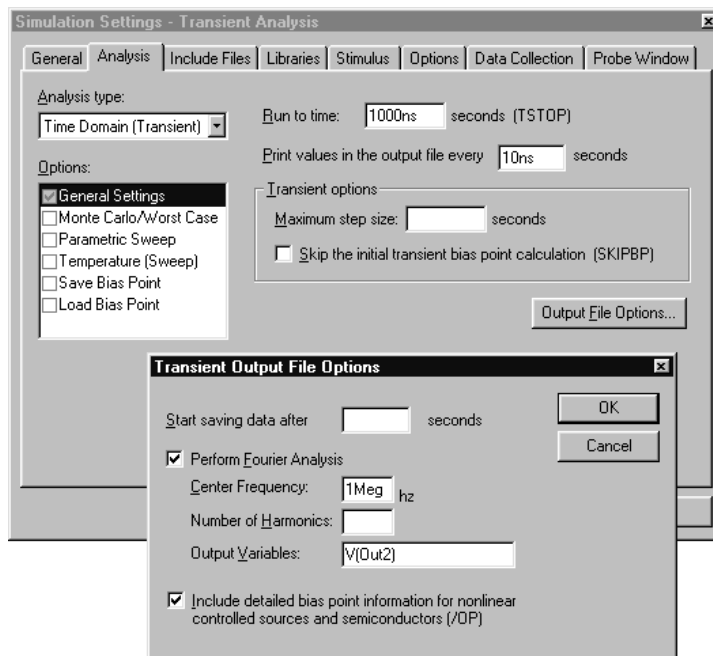


Figure 63 Transient analysis setup for EXAMPLE.OPJ.

During a transient analysis, any or all of the independent sources may have time-varying values. In EXAMPLE.OPJ, the only source which has a time-varying value is V1 (VSIN part) with attributes:

$$\begin{aligned}V_{OFF} &= 0\text{v} \\V_{AMPL} &= 0.1\text{v} \\FREQ &= 5\text{Meg}\end{aligned}$$

V1's value varies as a 5 MHz sine wave with an offset voltage of 0 volts and a peak amplitude of 0.1 volts. In general, more than one source has time-varying values.

The example circuit EXAMPLE.OPJ is provided with the OrCAD program installation.

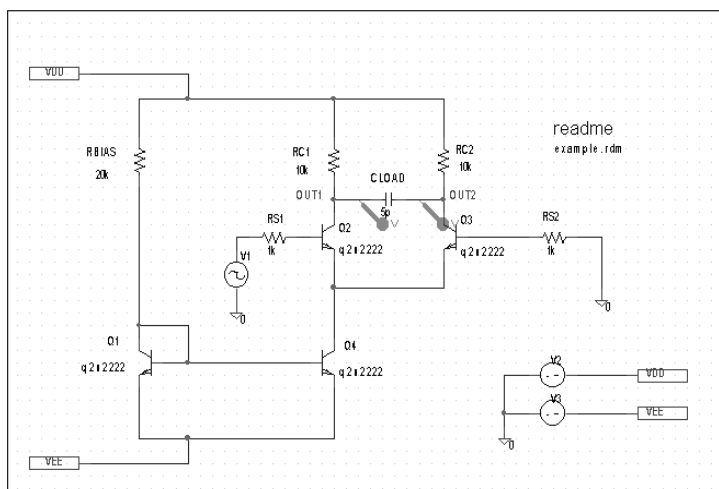


Figure 64 Example schematic EXAMPLE.OPJ.

The transient analysis does its own calculation of a bias point to start with, using the same technique as described for DC sweep. This is necessary because the initial values of the sources can be different from their DC values. To report the small-signal parameters for the transient bias point, use the Transient command and enable Detailed Bias Point. Otherwise, if you simply want the result of the transient run itself, you should only enable the Transient command.

In the simulation output file EXAMPLE.OUT, the bias-point report for the transient bias point is labeled INITIAL TRANSIENT SOLUTION.

Internal time steps in transient analyses

During analog analysis, PSpice maintains an internal time step which is continuously adjusted to maintain accuracy while not performing unnecessary steps. During periods of inactivity, the internal time step is increased. During active regions, it is decreased. The maximum internal step size can be controlled by specifying it in the Step Ceiling text box in the Transient dialog box. PSpice will never exceed either the step ceiling value or two percent of the total transient run time, whichever is less.

The internal time steps used may not correspond to the time steps at which information has been requested to be reported. The values at the print time steps are obtained by second-order polynomial interpolation from values at the internal steps.

Switching circuits in transient analyses

Running transient analysis on switching circuits can lead to long run times. PSpice must keep the internal time step short compared to the switching period, but the circuit's response extends over many switching cycles.

This technique is described in:

V. Bello, "Computer Program Adds SPICE to Switching-Regulator Analysis,"
Electronic Design, March 5, 1981.

One method of avoiding this problem is to transform the switching circuit into an equivalent circuit without switching. The equivalent circuit represents a sort of quasi steady-state of the actual circuit and can correctly model the actual circuit's response as long as the inputs do not change too fast.

Plotting hysteresis curves

Transient analysis can be used to look at a circuit's hysteresis. Consider, for instance, the circuit shown in Figure 65 (netlist in Figure 66).

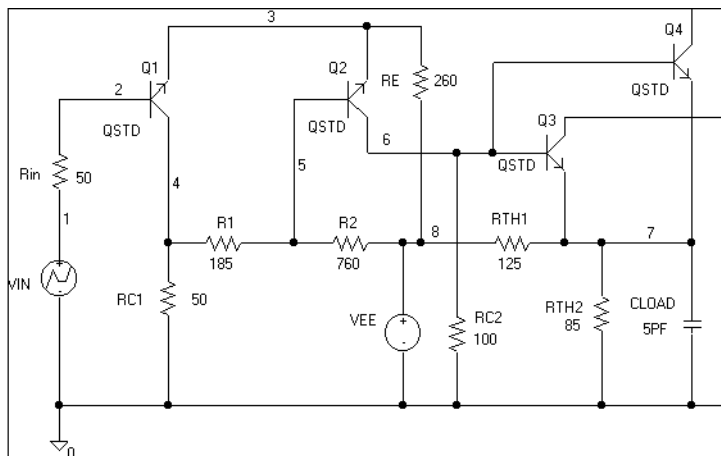


Figure 65 ECL-compatible Schmitt trigger.

```

* Capture Netlist

R_RIN      1 2 50
R_RC1      0 3 50
R_R1       3 5 185
R_R2       5 8 760
R_RC2      0 6 100
R_RE       4 8 260
R_RTH2     7 0 85
C_CLOAD    0 7 5PF
V_VEE      8 0 dc -5
V_VIN      1 0
+PWL 0 -8 1MS -1.0V 2MS -1.8V
R_RTH1     8 7 125
Q_Q1       3 2 4 QSTD
Q_Q2       6 5 4 QSTD
Q_Q3       0 6 7 QSTD
Q_Q4       0 6 7 QSTD

```

Figure 66 Netlist for Schmitt trigger circuit.

The QSTD model is defined as:

```

.MODEL QSTD NPN( is=1e-16 bf=50 br=0.1 rb=50 rc=10
tf=.12ns tr=5ns
+ cje=.4pF pe=.8 me=.4 cjc=.5pF pc=.8 mc=.333 ccs=1pF
va=50)

```

Instead of using the DC sweep to look at the hysteresis, use the transient analysis, (Print Step = .01ms and Final Time = 2ms) sweeping VIN from -1.8 volts to -1.0 volts and back down to -1.8 volts, very slowly. This has two advantages:

- it avoids convergence problems
- it covers both the upward and downward transitions in one analysis

After the simulation, in the Probe window in PSpice, the X axis variable is initially set to be Time. By selecting X Axis Settings from the Plot menu and clicking on the Axis Variable button, you can set the X axis variable to be V(1). Then use Add on the Trace menu to display V(7), and change the X axis to a user defined data range from -1.8V to -1.0V (Axis Settings on the Plot menu). This plots the output of the Schmitt trigger against its input, which is the desired outcome. The result looks similar to Figure 67.

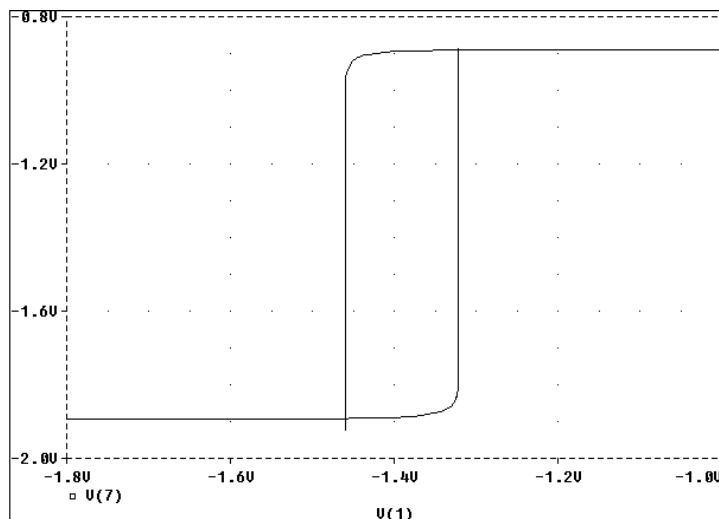


Figure 67 Hysteresis curve example: Schmitt trigger.

Fourier components

Note You must do a transient analysis in order to do a Fourier analysis. The sampling interval used during the Fourier transform is equal to the print step specified for the transient analysis.

Fourier analysis is enabled through the Output File Options dialog box under the Time Domain (Transient) Analysis type. Fourier analysis calculates the DC and Fourier components of the result of a transient analysis. By default, the first through ninth components are computed; however, more can be specified.

When selecting Fourier to run a harmonic decomposition analysis on a transient waveform, only a portion of the waveform is used. Using the Probe window in PSpice, a Fast Fourier Transform (FFT) of the complete waveform can be calculated and its spectrum displayed.

In the example shown in Figure 63 on page 10-263, the voltage waveform at node OUT2 from the transient analysis is to be used and the fundamental frequency is to be one megahertz for the harmonic decomposition. The period of fundamental frequency is one microsecond (inverse of the fundamental frequency). Only the last one microsecond of the transient analysis is used, and that

portion is assumed to repeat indefinitely. Since V1's sine wave does indeed repeat every one microsecond, this is sufficient. In general, however, you must make sure that the fundamental Fourier period fits the waveform in the transient analysis.

Parametric and temperature analysis

11

Chapter overview

This chapter describes how to set up parametric and temperature analyses. Parametric and temperature are both simple multi-run analysis types.

This chapter includes the following sections:

- [Parametric analysis on page 11-272](#)
- [Temperature analysis on page 11-281](#)

Parametric analysis

Minimum requirements to run a parametric analysis

Minimum circuit design requirements

- Set up the circuit according to the swept variable type as listed in [Table 1](#).
- Set up a DC sweep, AC sweep, or transient analysis.

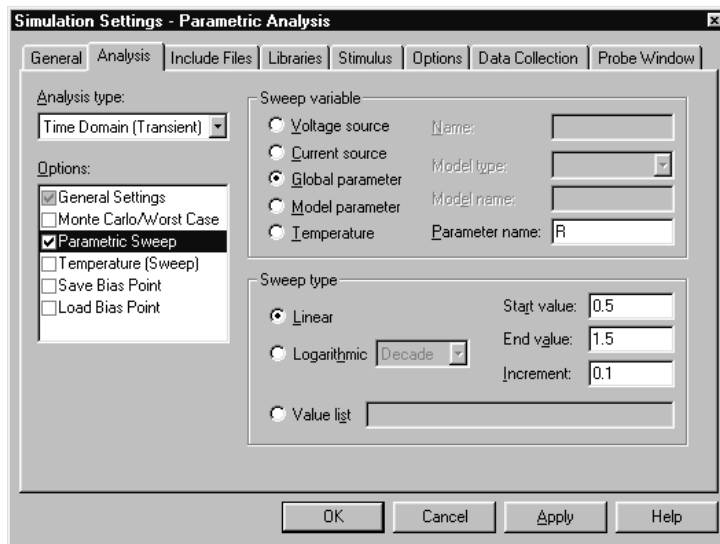
Table 1 *Parametric analysis circuit design requirements*

Swept variable type	Requirement
voltage source	voltage source with a DC specification (VDC, for example)
temperature	none
current source	current source with a DC specification (IDC, for example)
model parameter	PSpice model
global parameter	global parameter defined with a parameter block (PARAM)

Minimum program setup requirements

See [Setting up analyses on page 7-197](#) for a description of the Simulation Settings dialog box.

- 1 In the Simulation Settings dialog box, from the Analysis type list box, select Time Domain (Transient).
- 2 Under Options, select Parametric Sweep if it is not already enabled.
- 3 Specify the required parameters for the sweep.



- 4 Click OK to save the simulation profile.
- 5 From the PSpice menu, choose Run to start the simulation.

Note *Do not specify a DC sweep and a parametric analysis for the same variable.*

Overview of parametric analysis

Parametric analysis performs multiple iterations of a specified standard analysis while varying a global parameter, model parameter, component value, or operational temperature. The effect is the same as running the circuit several times, once for each value of the swept variable.

See [Parametric analysis on page 2-42](#) for a description of how to set up a parametric analysis.

RLC filter example

This example shows how to perform a parametric sweep and analyze the results with performance analysis.

Use performance analysis to derive values from a series of simulator runs and plot these values versus a parameter that varies between the simulator runs.

For this example, the derived values are the overshoot and the rise time versus the damping resistance of the filter.

Entering the design

The schematic representation for the RLC filter (RLCFILT.OPJ) is shown in Figure 68.

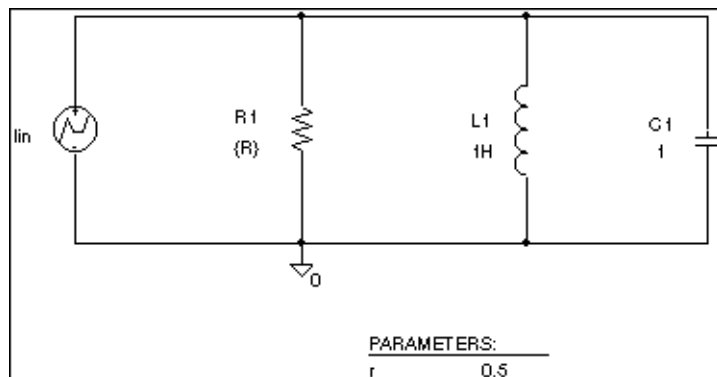


Figure 68 *Passive filter schematic.*

This series of PSpice runs varies the value of resistor R1 from 0.5 to 1.5 ohms in 0.1 ohm steps. Since the time-constant of the circuit is about one second, perform a transient analysis of approximately 20 seconds.

Create the circuit in OrCAD Capture by placing a piecewise linear independent current source (IPWL from SOURCE.OLB). Set the current source properties as follows:

AC = 1a
 T1 = 0s
 I1 = 0a
 T2 = 10ms

```

I2 = 0a
T3 = 10.1ms
I3 = 1a

```

Place an instance of a resistor and set its VALUE property to the expression, {R}. To define R as a global parameter, place a PARAM pseudocomponent and use the Property Editor to create a new property R and set its value to 0.5. Place an inductor and set its value to 1H, place a capacitor and set its value to 1, and place an analog ground symbol (0 from SOURCE.OLB). Wire the schematic symbols together as shown in Figure 68.

Running the simulation

Run PSpice with the following analyses enabled:

transient	print step:	100ms
	final time:	20s
parametric	swept var. type:	global parameter
	sweep type:	linear
	name:	R
	start value:	0.5
	end value:	1.5
	increment:	0.1

After setting up the analyses, start the simulation by choosing Run from the PSpice menu.

Using performance analysis to plot overshoot and rise time

After performing the simulation that creates the data file RLCFILT.DAT, you can calculate the specified performance analysis goal functions.

When the simulation is finished, a list appears containing all of the sections (runs) in the data file produced by PSpice. To use the data from every run, select All and click OK in the Available Selections dialog box. In the case of Figure 69, the trace I(L1) from the ninth section was added by specifying the following in the Add Traces dialog box:

```
I(L1)@9
```

To display the Add Traces dialog box, from the Trace menu, choose Add Trace or click the Add Trace toolbar button.



Troubleshooting tip

More than one PSpice run or data section is required for performance analysis. Because one data value is derived for each waveform in a related set of waveforms, at least two data points are required to produce a trace.

Use Eval Goal Function (from the Trace menu) to evaluate a goal function on a single waveform and produce a single data point result.



The genrise and overshoot goal functions are contained in the file PSPICE.PRB in the OrCAD directory.

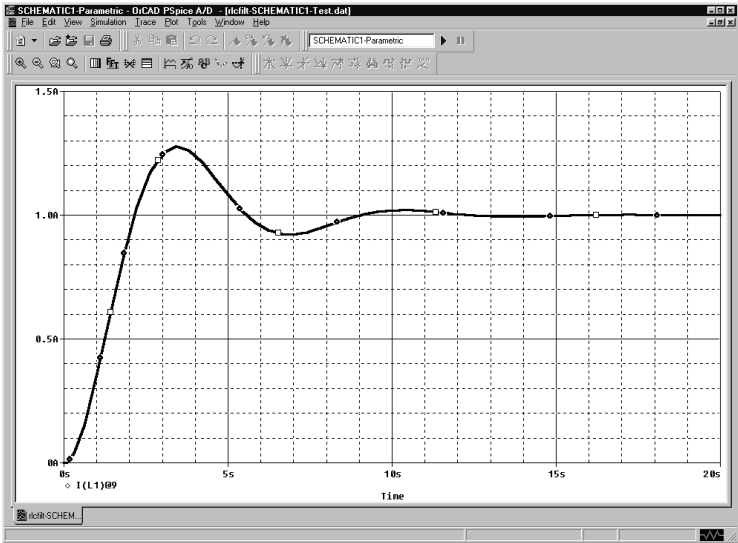


Figure 69 Current of L1 when R1 is 1.5 ohms.

To run performance analysis

- 1 From the Trace menu, choose Performance Analysis .
- 2 Click OK.

PSpice resets the X-axis variable for the graph to be the parameter that changed between PSpice runs. In the example, this is the R parameter.

To see the rise time for the current through the inductor L1, click the Add Trace toolbar button and then enter:

```
genrise( I(L1) )
```

Figure 70, shows how the rise time decreases as the damping resistance increases for the filter.

Another Y axis can be added to the plot for the overshoot of the current through L1 by selecting Add Y Axis from the Plot menu. The Y axis is immediately added. Now click the Add Trace toolbar button and enter:

```
overshoot( I(L1) )
```

Figure 70 shows how the overshoot increases with increasing resistance.

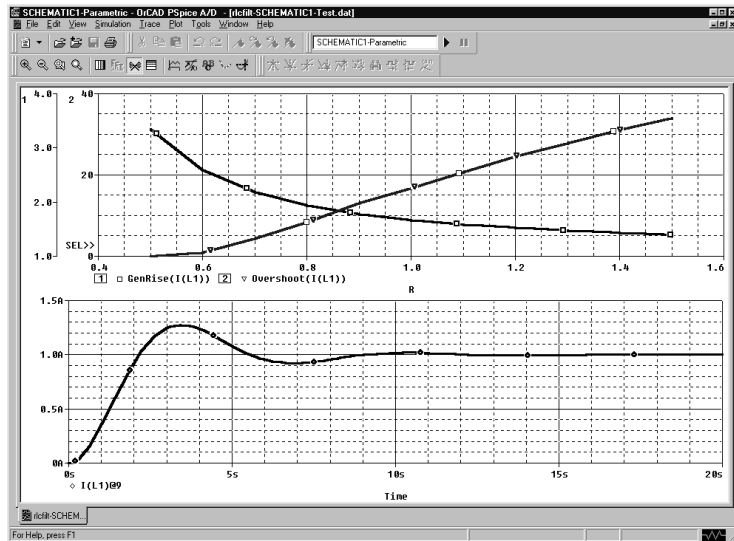


Figure 70 Rise time and overshoot vs. damping resistance.

This technique for measuring branch capacitances works well in both simple and complex circuits.

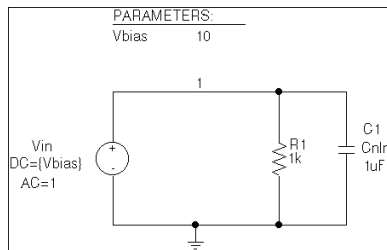


Figure 71 RLC filter example circuit.

Example: frequency response vs. arbitrary parameter

You can view a plot of the linear response of a circuit at a specific frequency as one of the circuit parameters varies (such as the output of a band pass filter at its center frequency vs. an inductor value).

In this example, the value of a nonlinear capacitance is measured using a 10 kHz AC signal and plotted versus its bias voltage. The capacitance is in parallel with a resistor, so a trace expression is used to calculate the capacitance from the complex admittance of the R-C pair.

Setting up the circuit

Enter the circuit in Capture as shown in Figure 71

To create the capacitor model in the schematic editor:

- 1 Place a CBREAK part.
- 2 Select it so that it is highlighted.
- 3 From the Edit menu, choose PSpice Model.
- 4 In the Model Text frame, enter the following:

```
.model Cnln CAP(C=1 VC1=-0.01 VC2=0.05)
```
- 5 From the File menu, choose Save.

Set up the circuit for a parametric AC analysis (sweep Vbias), and run PSpice. Include only the frequency of interest in the AC sweep.

To display the results

Use PSpice to display the capacitance calculated at the frequency of interest versus the stepped parameter.

- 1 Simulate the circuit.
- 2 Load all AC analysis sections.
- 3 From the Trace menu, choose Add Trace or click the Add Trace toolbar button.
- 4 Add the following trace expression:



```
IMG(-I(Vin)/V(1,0))/(2*3.1416*Frequency)
```

Or add the expression:

```
CvF(-I(Vin)/V(1,0))
```

Where CvF is a macro which measures the effective capacitance in a complex conductance. Macros are defined using the Macros command on the Trace menu. The CvF macro should be defined as:

```
CvF(G)= IMG(G)/(2*3.1416*Frequency)
```

Note *-I(Vin)/V(1) is the complex admittance of the R-C branch; the minus sign is required for correct polarity.*

To use performance analysis to plot capacitance vs. bias voltage

- 1 From the Trace menu, choose Performance Analysis.
- 2 Click Wizard.
- 3 Click Next>.
- 4 Click YatX in the Choose a Goal Function list, and then click Next>.
- 5 In the Name of Trace text box, type the following:

```
CvF(-I(Vin)/V(1))
```
- 6 In the X value to get Y value at text box, type 10K.
- 7 Click Next>.

The wizard displays the gain trace for the first run to text the goal function (YatX).

- 8 Click Finish.

The resultant plot is shown in Figure 72.

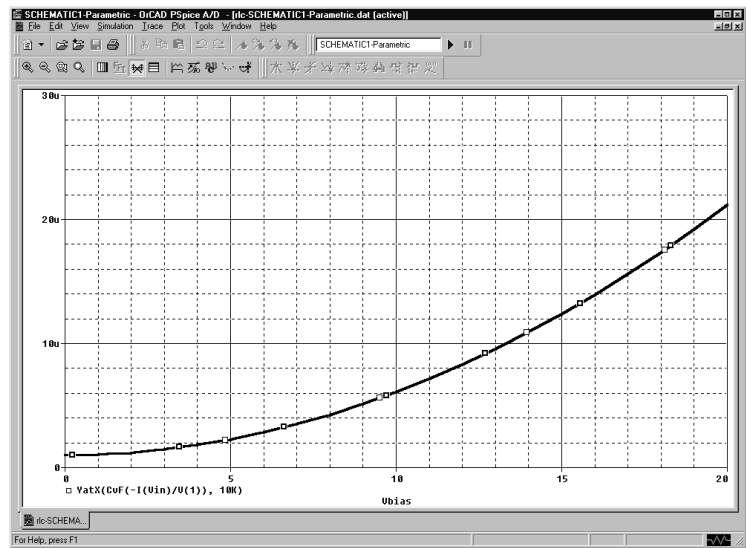


Figure 72 Plot of capacitance versus bias voltage.

Temperature analysis

Minimum requirements to run a temperature analysis

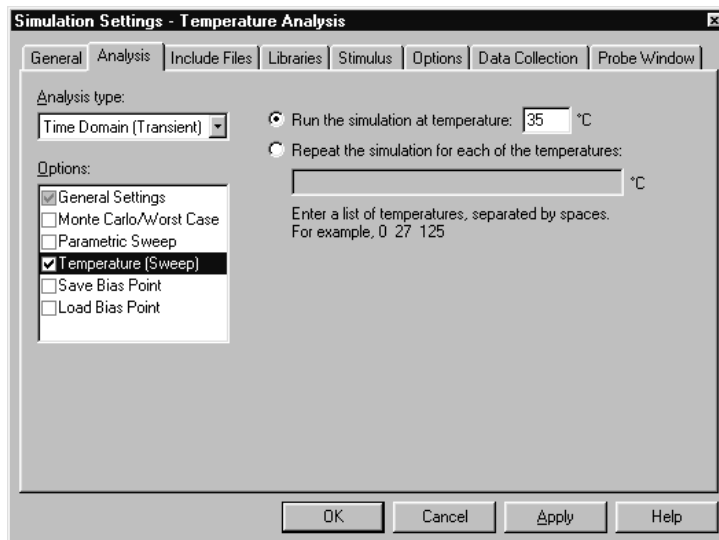
Minimum circuit design requirements

None.

Minimum program setup requirements

- 1 In the Simulation Settings dialog box, from the Analysis type list box, select Time Domain (Transient).
- 2 Under Options, select Temperature Sweep if it is not already enabled.
- 3 Specify the required parameters for the sweep.

See [Setting up analyses on page 7-197](#) for a description of the Simulation Settings dialog box.



- 4 Click OK to save the simulation profile.
- 5 From the PSpice menu, choose Run to start the simulation.

Running multiple analyses for different temperatures can also be achieved using parametric analysis (see [Parametric analysis on page 11-272](#)). With parametric analysis, the temperatures can be specified either by list, or by range and increments within the range.

Overview of temperature analysis

For a temperature analysis, PSpice reruns standard analyses set in the Simulation Settings dialog box at different temperatures.

You can specify zero or more temperatures. If no temperature is specified, the circuit is run at 27°C. If more than one temperature is listed, the simulation runs once for each temperature in the list.

Setting the temperature to a value other than the default results in recalculating the values of temperature-dependent devices. In EXAMPLE.OPJ (see Figure 73), the temperature for all of the analyses is set to 35°C. The values for resistors RC1 and RC2 are recomputed based upon the CRES model which has parameters TC1 and TC2 reflecting linear and quadratic temperature dependencies.

Likewise, the Q3 and Q4 device values are recomputed using the Q2N2222 model which also has temperature-dependent parameters. In the simulation output file, these recomputed device values are reported in the section labeled TEMPERATURE ADJUSTED VALUES.

The example circuit EXAMPLE.OPJ is provided with the OrCAD program installation.

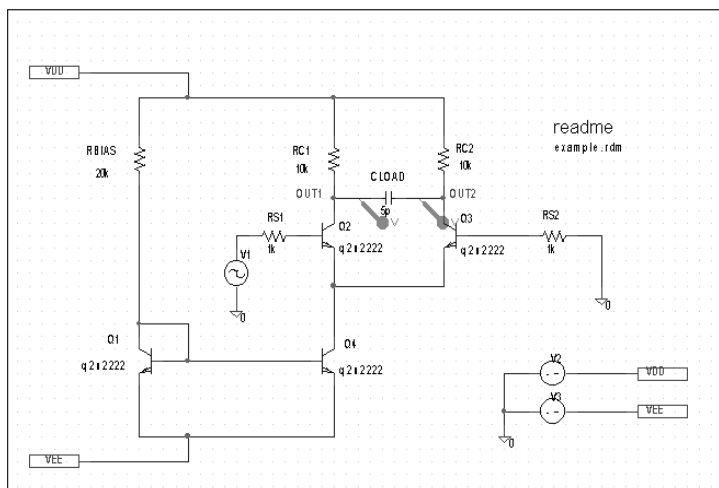


Figure 73 Example schematic EXAMPLE.OPJ.

Monte Carlo and sensitivity/ worst-case analyses

12

Chapter overview

This chapter describes how to set up Monte Carlo and sensitivity/worst-case analyses and includes the following sections:

- [Statistical analyses on page 12-284](#)
- [Monte Carlo analysis on page 12-289](#)
- [Worst-case analysis on page 12-306](#)

Statistical analyses

Monte Carlo and sensitivity/ worst-case are statistical analyses. This section describes information common to both types of analyses.

See [Monte Carlo analysis on page 12-289](#) for information specific to Monte Carlo analyses, and see [Worst-case analysis on page 12-306](#) for information specific to sensitivity/ worst-case analyses.

Overview of statistical analyses

Generating statistical results

As the number of Monte Carlo or worst-case runs increases, simulation takes longer and the data file gets larger. Large data files may be slow to open and slow to draw traces.

One way to work around this is to set up an overnight batch job to run the simulation and execute commands. You can even set up the batch job to produce a series of plots on paper to be ready for you in the morning.

The Monte Carlo and worst-case analyses vary the lot or device tolerances of devices between multiple runs of an analysis (DC, AC, or transient). Before running the analysis, you must set up the model and/or lot tolerances of the model parameter to be investigated.

A Monte Carlo analysis performs a Monte Carlo (statistical) analysis of the circuit. A worst-case analysis performs a sensitivity and worst-case analysis of the circuit.

Sensitivity/ worst-case analyses are different from Monte Carlo analyses in that they compute the parameters using the sensitivity data rather than using random numbers.

You can run either a Monte Carlo or a worst-case analysis, but you *cannot* run both at the same time. Multiple runs of the selected analysis are done while parameters are varied. You can select only one analysis type (AC, DC, or transient) per run. The selected analysis is repeated in subsequent passes of the analysis.

Output control for statistical analyses

Monte Carlo and sensitivity/worst-case analyses generate the following types of reports:

- Model parameter values used for each run (that is, the values with tolerances applied)
- Waveforms from each run, as a function of specifying data collection, or by specifying output variables in the analysis set up
- Summary of all the runs using a collating function

Output is saved to the data file for use by the waveform analyzer. For Monte Carlo analyses, you can use the performance analysis feature to produce histograms of derived data.

For information about performance analysis, see [RLC filter example on page 11-274](#).

For information about histograms, see [Creating histograms on page 12-303](#).

Model parameter values reports

To produce a list of the model parameters actually used for each run,

- 1 In the Simulation Settings dialog box, click the Analysis tab.
- 2 From the Analysis type list, select an analysis type.
- 3 Under Options, select Monte Carlo/Worst Case.
- 4 Click the More Settings button.
- 5 Select List model parameter values.
- 6 Click OK to close the Simulation Settings dialog box.

This list is written to the simulation output file at the beginning of the run and contains the parameters for each device, as opposed to the parameters for each .MODEL statement. This is because devices can have different parameter values when using a model statement containing a DEV tolerance.

Note that for midsize and large circuits, the List option can produce a large output file.

Waveform reports

For Monte Carlo analyses, there are five variations of the output that you can specify in the Save data from text box on the Monte Carlo dialog box. Options:

In excess of about 10 runs, the waveform display can look more like a band than a set of individual waveforms. This can be useful for seeing the typical spread for a particular output variable. As the number of runs increases, the spread more closely approximates the actual worst-case limits for the circuit.

<none>	No output is generated
All	Forces all output to be generated (including nominal run)
First*	Generates output only during the first n runs
Every*	Generates output for every n th run
Runs(list)*	Does specified analysis and generates outputs only for the listed runs (up to 25 values can be specified in the list)

The * indicates that you can set the number of runs in the runs text box.

Values for the output variables specified in the selected analyses are saved to the simulation output file and data file.

Note *Even a modest number of runs can produce large output files.*

Collating functions

You can further compress the results of Monte Carlo and worst-case analyses. If you use the collating function, a single number represents each run. (Click the Output File Options button and select a function from the Find list.) A table of deviations per run is reported in the simulation output file.

Collating functions are listed in [Table 1](#).

Table 1 *Collating functions used in statistical analyses*

Function	Description
YMAX	Find the greatest difference in each waveform from the nominal
MAX	Find the maximum value of each waveform
MIN	Find the minimum value of each waveform
RISE_EDGE	Find the first occurrence of the waveform crossing above a specified threshold value
FALL_EDGE	Find the first occurrence of the waveform crossing below a specified threshold value

Refer to *Temperature Effects on Monte Carlo Analysis* in the *Application Notes* manual for more information.

Temperature considerations in statistical analyses

The statistical analyses perform multiple runs, as does the temperature analysis. Conceptually, the Monte Carlo and worst-case loops are inside the temperature loop.

However, since both temperature and tolerances affect the model parameters, OrCAD recommends not using temperature analysis when using Monte Carlo or worst-case analysis.

Also, you cannot sweep the temperature in a DC sweep analysis or put tolerances on temperature coefficients while performing one of these statistical analyses. In EXAMPLE.DSN, the temperature value is fixed at 35°C.

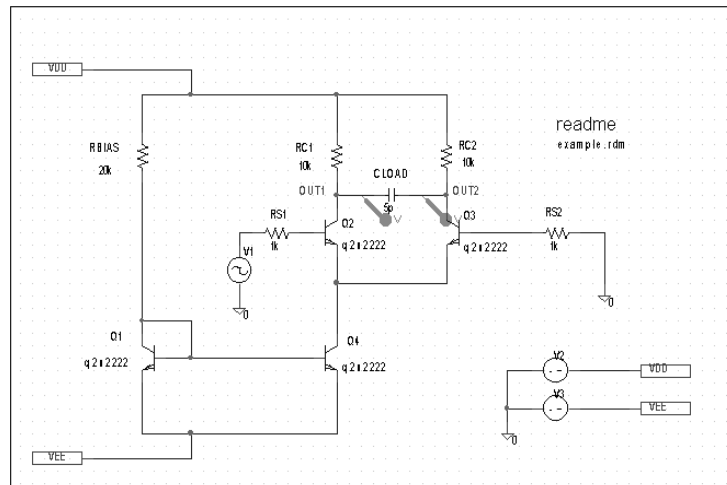


Figure 74 *Example schematic EXAMPLE.DSN.*

The example schematic EXAMPLE.DSN is provided on the OrCAD installation CD.

Monte Carlo analysis

The Monte Carlo analysis calculates the circuit response to changes in part values by randomly varying all of the model parameters for which a tolerance is specified. This provides statistical data on the impact of a device parameter's variance.

With Monte Carlo analysis, model parameters are given tolerances, and multiple analyses (DC, AC, or transient) are run using these tolerances.

Monte Carlo analysis is frequently used to predict yields on production runs of a circuit.

For EXAMPLE.DSN in Figure 74 on page 12-288, you can analyze the effects of variances in the values of resistors RC1 and RC2 by assigning a model description to these resistors that includes a 5% device tolerance on the multiplier parameter R.

Then you can perform a Monte Carlo analysis. First, the simulator performs a DC analysis with the nominal R multiplier value for RC1 and RC2. Then it performs a set number of additional runs with the R multiplier varied independently for RC1 and RC2 within a 5% tolerance.

To modify example.dsn and set up simulation

- 1 Replace RC1 and RC2 with RBREAK parts, setting property values to match the resistors that are being replaced (VALUE=10k) and reference designators to match previous names.
- 2 Select PSpice Model from the Edit menu. Create the model CRES as follows:

```
.MODEL CRES RES( R=1 DEV=5% TC1=0.02
+ TC2=0.0045 )
```

From the File menu, choose Save. By default, Capture saves the definition to the model library EXAMPLE.LIB and automatically configures the file for local use with the current schematic.

- 3 In Capture, set up a new Monte Carlo analysis as shown in Figure 75. The analysis specification tells PSpice to do one nominal run and four Monte Carlo

TC1 is the linear temperature coefficient.
TC2 is the quadratic temperature coefficient.

runs, saving the DC analysis output from those five runs.

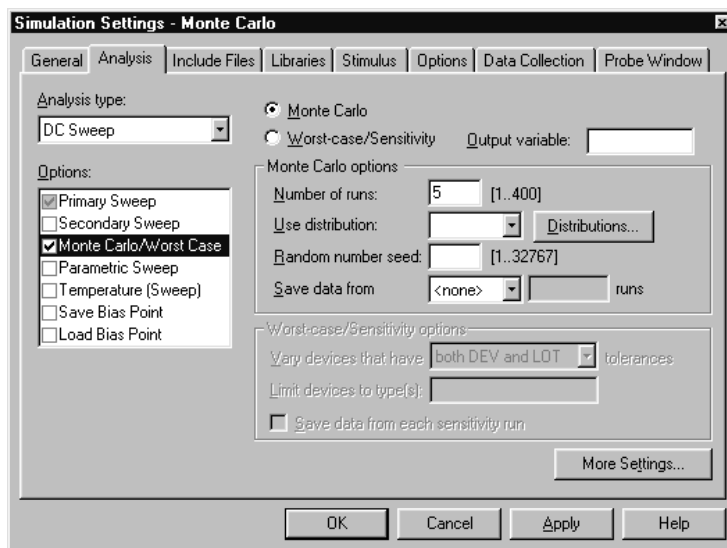


Figure 75 Monte Carlo analysis setup for EXAMPLE.DSN.

PSpice starts by running *all* of the analyses enabled in the Simulation Settings dialog box with all parameters set to their nominal values.

However, with Monte Carlo enabled, PSpice saves the DC sweep analysis results for later reference and comparison. After the nominal analyses are finished, PSpice performs the additional specified analysis runs (in this example, DC sweep).

Subsequent runs use the same analysis specification as the nominal run with one major exception: instead of using the nominal parameter values, the tolerances are applied to set new parameter values and thus, new part values.

There is a trade-off in choosing the number of Monte Carlo runs. More runs provide better statistics, but they require more time. The amount of time scales directly with the number of runs: 20 transient analyses take 20 times as long as one transient analysis. During Monte Carlo runs, the PSpice status display includes the current run number and the total number of runs left.

PSpice offers a facility to generate histograms of data derived from Monte Carlo waveform families through the performance analysis feature.

For information about performance analysis, see [RLC filter example on page 11-274](#).

For information about histograms, see [Creating histograms on page 12-303](#).

Reading the summary report

The summary report generated in this example (see Figure 76) specifies that the waveform generated from V(OUT1, OUT2) should be the subject of the collating function YMAX. In each of the last four runs, the new V(OUT1, OUT2) waveform is compared to the nominal V(OUT1, OUT2) waveform for the first run, calculating the maximum deviation in the Y direction (YMAX collating function). The deviations are printed in order of size along with their run number .

```

****      SORTED DEVIATIONS OF V(OUT1,OUT2) TEMPERATURE =   35.000 DEG C

              MONTE CARLO SUMMARY

*****

Mean Deviation =   -.2477
Sigma           =    .3035

  RUN          MAX DEVIATION FROM NOMINAL
Pass   3          .5729 (1.89 sigma) lower  at U_U1 =   -.02
        ( 94.885% of Nominal)
Pass   4          .3549 (1.17 sigma) lower  at U_U1 =   -.02
        ( 96.832% of Nominal)
Pass   2          .3122 (1.03 sigma) lower  at U_U1 =   -.02
        ( 97.212% of Nominal)
Pass   5          .2493 ( .82 sigma) higher  at U_U1 =   -.02
        (102.23% of Nominal)

```

Figure 76 Summary of Monte Carlo runs for *EXAMPLE.OPJ*.

With the List option enabled, a report is also generated showing the parameter value used for each device in each run. In this case (see Figure 77), run three shows the highest deviation.

```
* C:\ORCAD\EX\EXAMPLE.SCH

****      UPDATED MODEL PARAMETERS      TEMPERATURE =  35.000 DEG C

      MONTE CARLO PASS      3

*****

**** CURRENT MODEL PARAMETERS FOR DEVICES REFERENCING cres
      R_RC1      R_RC2
R      1.0304E+00      9.5053E-01
```

Figure 77 *Parameter values for Monte Carlo pass three.*

Example: Monte Carlo analysis of a pressure sensor

This example shows how the performance of a pressure sensor circuit with a pressure-dependent resistor bridge is affected by manufacturing tolerances, using Monte Carlo analysis to explore these effects.

Drawing the schematic

To begin, construct the bridge as shown in Figure 78.

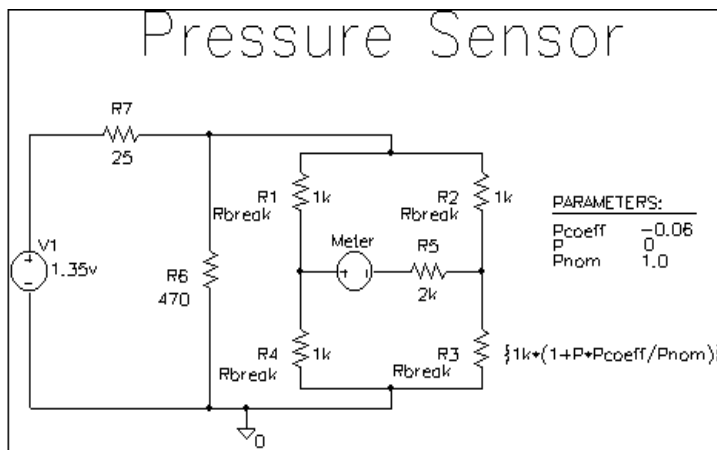


Figure 78 Pressure sensor circuit.

Here are a few things to know when placing and connecting the part:

- To get the part you want to place, from the Place menu, choose Part.
- To rotate a part before placing it, press **[R]**.
- For V1 and Meter, place a generic voltage source using the VSRC part. When you place the source for the meter, change its name by double-clicking the part and typing `Meter` in the Reference cell in the Parts Spreadsheet.
- For R1-R7, place a resistor using the R part.
- Place the analog ground using the 0 ground symbol.



- To connect the parts, from the Place menu, choose Wire.
- To move values or reference designators, click the value or reference designator to select it, then drag it to the new location.

Defining part values

Define the part values as shown in Figure 78. For the pressure sensor, you need to do the following:

- Change the resistor values for R3, R5, R6, and R7 from their default value of 1 k.
- Set the DC value for the V1 voltage source.

Note Because the Meter source is used to measure current, it has no DC value and can be left unchanged.

To change resistor values

- 1 Double-click the value for a resistor.
- 2 Type the new value. Depending on the resistor you are changing, set its value to one of the following (refer to Figure 78).

Table 1

If you are changing this resistor...	Type this...
R3	{1k*(1+P*Pcoeff/Pnom)}
R5	2k
R6	470
R7	25

Note The value for R3—{1k*(1+P*Pcoeff/Pnom)}—is an expression that represents linear dependence of resistance on pressure. To complete the definition for R3, you will create and define global parameters for Pcoeff, P, and Pnom later on in this example

- 3 Repeat steps 1-2 for each resistor on your schematic page.

To set the DC value for the V1 source and make it visible

- 1 Double-click the V1 source part.
- 2 In the Parts Spreadsheet, click in the cell under the DC column.
- 3 Type 1.35v.

- 4 Click the Display button.
- 5 In the Display Format frame, choose the Value Only option to make the DC value (1.35V) visible on the schematic.
- 6 Click OK, then click Apply to apply the changes you have made to the part.
- 7 Close the Parts Spreadsheet.

Setting up the parameters

To complete the value specification for R3, define the global parameters Pcoeff, P, and Pnom.

To define and initialize Pcoeff, P, and Pnom

- 1 Place a PARAM part on the schematic page.
- 2 Double-click the PARAM part to display the Parts Spreadsheet.
- 3 For each parameter, create a new property by clicking New and typing its name. Enter its corresponding value by clicking in the cell under the new property name and typing its value. Specify the parameter name and corresponding value as follows.

Table 2

Property	Value
Pcoeff	-0.06
P	0
Pnom	1.0

- 4 Click Apply to save the changes you have made then close the Parts Spreadsheet.

When PSpice runs a Monte Carlo analysis, it uses tolerance values to determine how to vary model parameters during the simulation.

Using resistors with models

To explore the effects of manufacturing tolerances on the behavior of this circuit, you set device (DEV) and (LOT) tolerances on the model parameters for resistors R1, R2, R3, and R4 in a later step (see page [12-297](#)). This means you need to use resistor parts that have model associations.

Because R parts do not have associated models (and therefore no model parameters), change the resistor parts to Rbreak parts that do have models.

To replace R1, R2, R3, and R4 with the RBREAK part

- 1 Click R1 to select it.
- 2 Hold down the **[Ctrl]** key and click R2, R3 and R4 to add them to the selection set.
- 3 Press **[Delete]** to delete the selection set.
- 4 From the Place menu, choose Part.
- 5 Type **RBREAK** in the Part text box. (If RBREAK is not available, click the Add Library button and select BREAKOUT.OLB to configure it for use in Capture.)
- 6 Click OK.
- 7 Manually place the RBREAK part in the circuit diagram where R1, R2, R3 and R4 were located.
- 8 Double-click on each RBREAK part and change the reference designators as desired.

Saving the design

Before editing the models for the Rbreak resistors, save the schematic.

To save the design

- 1 From Capture's File menu, choose Save.

Defining tolerances for the resistor models

This section shows how to assign device (DEV) and lot (LOT) tolerances to the model parameters for resistors R1, R2, R3, and R4 using the model editor.

To assign 2% device and 10% lot tolerances to the resistance multiplier for R1

- 1 Select R1.
- 2 From the Edit menu, choose PSpice Model.
Capture searches the libraries for the Rbreak model definition and makes a copy to create an instance model.
- 3 To change the instance model name from Rbreak to Rmonte1, do the following:
 - a In the Model Text frame, double-click Rbreak.
 - b Type RMonte1.
- 4 To add a 2% device tolerance and a 10% lot tolerance to the resistance multiplier, do the following:
 - a Add the following to the .MODEL statement (after R=1):

```
DEV=2% LOT=10%
```

The model editing window should look something like Figure 79.

- 5 From the File menu, choose Save.

By default, Capture saves the RMonte1 .MODEL definition to the design_name.lib library, which is

You can use the model editor to change the .MODEL or .SUBCKT syntax for a model definition. To find out more about the model editor, see [Editing model text on page 4-110](#), or refer to the online *PSpice Reference Manual*.

To find out more about adding model libraries to the configuration, see [Configuring model libraries on page 4-100](#).

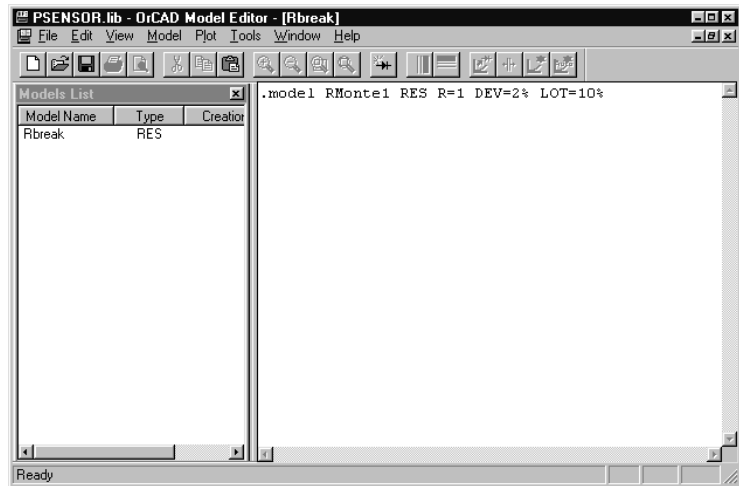


Figure 79 Model definition for RMonte1.

PSENSOR.LIB. Capture also automatically configures the library for local use.

To have resistors R2 and R4 use the same tolerances as R1

- 1 In Capture's schematic page editor, select R2 and R4.
- 2 From the Edit menu, select Properties.
- 3 In the R2 row, click in the cell under the Implementation column and type RMonte1.
- 4 In the R4 row, click in the cell under the Implementation column and type RMonte1.

To assign 5% device tolerance to the resistance multiplier for R3

- 1 Select R3.
- 2 From the Edit menu, select PSpice Model.
- 3 In the Model Text frame, change the .MODEL statement to:

```
.model RTherm RES R=1 DEV=5%
```

- 4 From the File menu, choose Save.

Your schematic page should look like Figure 80.

Setting up the analyses

This section shows how to define and enable a DC analysis that sweeps the pressure value and a Monte Carlo analysis that runs the DC sweep with each change to the resistance multipliers.

To set up the DC sweep

- 1 In the PSpice menu, choose New Simulation Profile or Edit Simulation Settings. (If this is a new simulation, enter the name of the profile and click OK.)

See [Setting up analyses on page 7-197](#) for a description of the Simulation Settings dialog box.

The Simulation Settings dialog box appears.

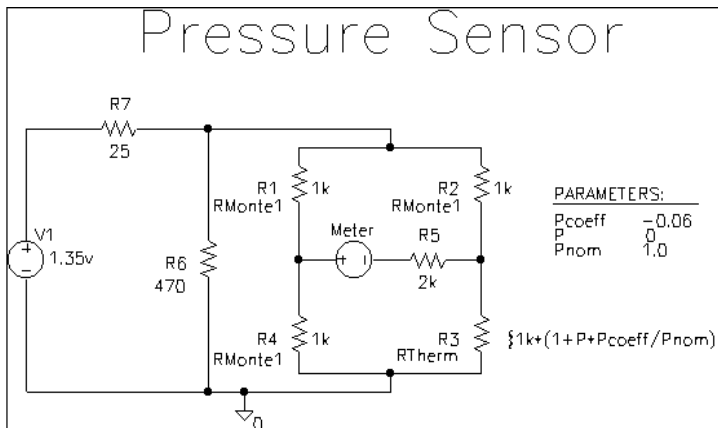


Figure 80 Pressure sensor circuit with RMonte1 and RTherm model definitions.

- 2 Select DC Sweep in the Analysis type list box.
- 3 In the Sweep Variable frame, select Global Parameter.
- 4 Enter the following values:

Table 3

In this text box...	Type this...
Parameter name	P
Start value	0
End value	5 . 0
Increment	0 . 1

To set up the Monte Carlo analysis

- 1 Select the Monte Carlo/Worst Case option.
- 2 Check Monte Carlo if it is not already selected.
- 3 In the Number of runs text box, type 10.
- 4 In the Save data from list box, select All.
- 5 Type I(Meter) in the Output variable text box.
- 6 Click OK to save the simulation profile.

Running the analysis and viewing the results

To complete setup, simulate, and view results

- 1 From Capture’s PSpice menu, choose Run to start the simulation

When the simulation is complete, PSpice automatically displays the selected waveform. Because PSpice ran a Monte Carlo analysis, it saved multiple runs or sections of data. These are listed in the Available Sections dialog box.
- 2 From PSpice’s Trace menu, choose Performance Analysis.
- 3 Click the Select sections button.
- 4 In the Available Sections dialog box, click the All button.
- 5 Click OK.

- 6 To display current through the Meter voltage source, do the following:
 - a From Capture's PSpice menu, point to markers and choose Current into Pin.
 - b Place a current probe on the left-hand pin of the Meter source.
- 7 Switch to the Probe window to see the family of curves for I(Meter) as a function of P.

Note *For more on analyzing Monte Carlo results in PSpice, see the next section on Monte Carlo histograms.*

Another way to view the family of curves without using schematic markers is as follows:

- 1 From PSpice's Trace menu, choose Add Trace.
- 2 In the Simulation Output Variables list, double-click I(Meter).

Monte Carlo Histograms

You can display data derived from Monte Carlo waveform families as histograms. This is part of the performance analysis feature.

In this example, you simulate a fourth-order Chebyshev active filter, running a series of 100 AC analyses while randomly varying resistor and capacitor values for each run. Then, having defined performance analysis goal functions for bandwidth and center frequency, you observe the statistical distribution of these quantities for the 100 runs.

Monte Carlo analysis is frequently used to predict yields on production runs of a circuit.

For more information about performance analysis, see [RLC filter example on page 11-274](#).

Chebyshev filter example

The Chebyshev filter is designed to have a 10 kHz center frequency and a 1.5 kHz bandwidth. The schematic page for the filter is shown in Figure 81. The stimulus specifications for V1, V2, and V3 are:

V1: DC=-15
 V2: DC=+15
 V3: AC=1

The parts are rounded to the nearest available 1% resistor and 5% capacitor value. In this example, note how the

bandwidth and the center frequency vary when 1% resistors and 5% capacitors are used in the circuit.

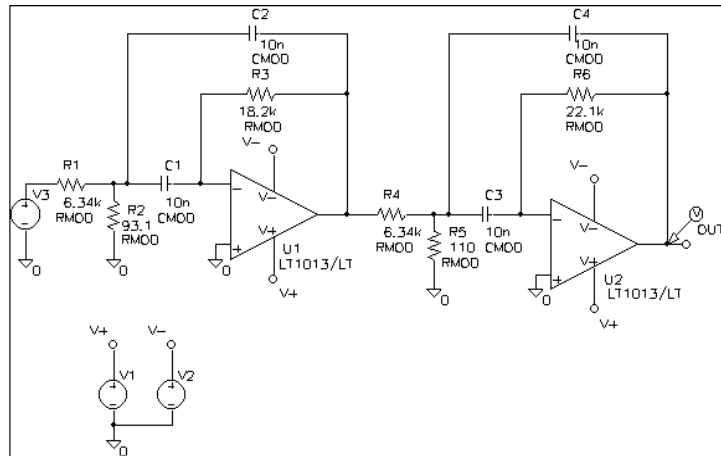


Figure 81 *Chebyshev filter.*

Creating models for Monte Carlo analysis

To vary the resistors and capacitors in the filter circuit, create models for these parts on which you can set device tolerances for Monte Carlo analysis. The BREAKOUT.OLB library contains generic devices for this purpose. The resistors and capacitors in this schematic are the Rbreak and Cbreak parts from BREAKOUT.OLB.

Using the Model Editor, modify the models for these parts as follows:

```
.model RMOD RES(R=1 DEV=1%)
.model CMOD CAP(C=1 DEV=5%)
```

Setting up the analysis

To analyze the filter, set up both an AC analysis and a Monte Carlo analysis. The AC analysis sweeps 50 points per decade from 100 Hz to 1 MHz. The Monte Carlo analysis is set to take 100 runs. The analysis type is AC and the output variable is V(OUT).

To set up the analysis

- 1 From PSpice's Trace menu, choose Performance Analysis.
- 2 In the Save data from list box, choose All.
- 3 Click OK.

Creating histograms

Because the data file can become quite large when running a Monte Carlo analysis, to view just the output of the filter, you place a voltage probe at the output of the filter.

To collect data for the marked node only

- 1 From the PSpice menu, choose New Simulation Profile or Edit Simulation Settings from the PSpice menu. (If this is a new simulation, enter the name of the profile and click OK.)

The Simulation Settings dialog box appears.

- 2 On the Data Collection tab, choose the At Probes only option.
- 3 Click OK.

To run the simulation and load Probe with data

- 1 From Capture's PSpice menu, choose Run to start the simulation.

When the simulation is complete, PSpice automatically displays the selected waveform. Because PSpice ran a Monte Carlo analysis, it saved multiple runs or sections of data. These are listed in the Available Sections dialog box.

- 2 From PSpice's Trace menu, choose Performance Analysis.
- 3 Click the Select sections button.
- 4 In the Available Sections dialog box, click All.

For information about performance analysis, see [RLC filter example on page 11-274](#).

You can also display this histogram by using the performance analysis wizard to display Bandwidth (VDB(OUT) , 1).

- 5 Click OK.

To display a histogram for the 1 dB bandwidth

- 1 From PSpice's Plot menu, choose Axis Settings.
- 2 Select the X Axis tab.
- 3 In the Processing Options frame, select the Performance Analysis check box.

- 4 Click OK.

The histogram display appears. The Y axis is the percent of samples.

- 5 From the Trace menu, choose Goal Functions.
- 6 Choose Bandwidth.
- 7 Click Eval.
- 8 Enter `VDB(OUT)` in the Name of trace to search text box.
- 9 Enter 1 in the db level down for bandwidth calc text box.
- 10 Click OK, then click Close to view the histogram.

To change the number of histogram divisions

- 1 From the Tools menu, choose Options.
- 2 In the Number of Histogram Divisions text box, replace 10 with 20.
- 3 Click OK.

The histogram for 1 dB bandwidth is shown in Figure 82.

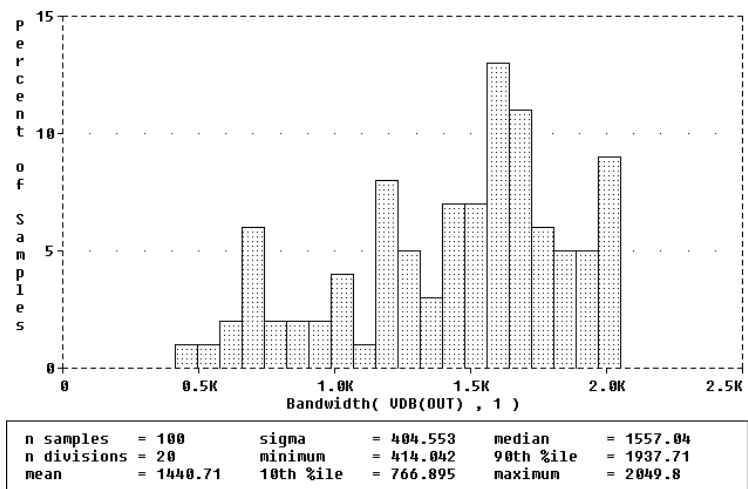


Figure 82 1 dB bandwidth histogram.

The statistics for the histogram are shown along the bottom of the display. The statistics show the number of Monte Carlo runs, the number of divisions or vertical bars that make up the histogram, mean, sigma, minimum, maximum, 10th percentile, median, and 90th percentile.

You can also show the distribution of the center frequency of the filter.

To display the center frequency

- 1 From the Trace menu, choose Goal Functions.
- 2 Choose CenterFreq.
- 3 Click Eval.
- 4 Enter `VDB(OUT)` in the Name of trace to search text box.
- 5 Enter 1 in the db level down for measurement text box.
- 6 Click OK, then click Close to view the histogram.

The new histogram replaces the previous histogram. To display both histograms at once, choose Add Plot to Window on the Plot menu before choosing Add from the Trace menu. The histogram of the center frequency is as shown in Figure 83.

If needed, you can turn off the statistical data display as follows:

- 1 From the Tools menu, choose Options.
- 2 Clear the Display Statistics check box.
- 3 Click Save, and then OK.

Ten percent of the goal function values is less than or equal to the 10th percentile number, and 90% of the goal function values is greater than or equal to that number.

If there is more than one goal function value that satisfies this criteria, then the 10th percentile is the midpoint of the interval between the goal function values that satisfy the criteria. Similarly, the median and 90th percentile numbers represent goal function values such that 50% and 90% (respectively) of the goal function values are less than or equal to those numbers.

Sigma is the standard deviation of the goal function values.

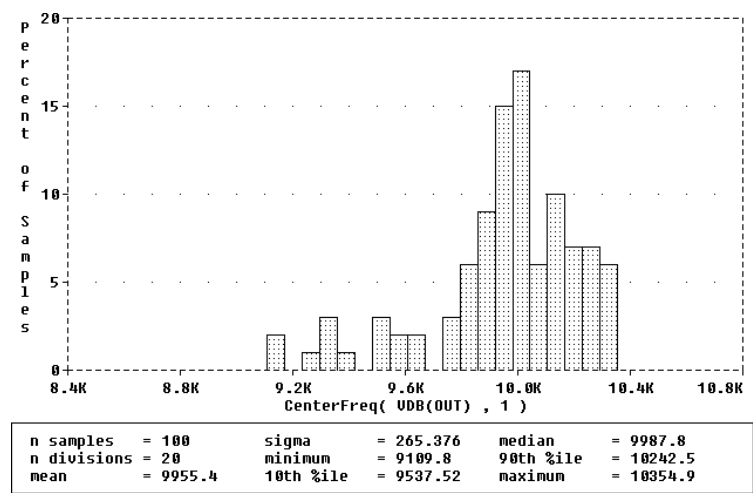


Figure 83 Center frequency histogram.

Worst-case analysis

This section discusses the analog worst-case analysis feature of PSpice. The information provided in this section explains how to use worst-case analysis properly and with realistic expectations.

Overview of worst-case analysis

Worst-case analysis is used to find the worst probable output of a circuit or system given the restricted variance of its parameters. For instance, if the values of R1, R2, and R3 can vary by $\pm 10\%$, then the worst-case analysis attempts to find the combination of possible resistor values which result in the worst simulated output. As with any other analysis, there are three important parts: inputs, procedure, and outputs.

Inputs

In addition to the circuit description, you need to provide two pieces of information:

- the parameter tolerances
- a definition of what *worst* means

You can set tolerances on any number of the parameters that characterize a model.

The criterion for determining the *worst* values for the relevant model parameters is defined in the .WC statement as a function of any standard output variable in a specified range of the sweep.

In a given range, reduce the measurement to a single value by one of these five collating functions:

MAX	Maximum output variable value
MIN	Minimum output variable value
YMAX	Output variable value at the point where it differs the most with the nominal run
RISE_EDGE (value)	Sweep value where the output variable value crosses <i>above</i> a given threshold value
FALL_EDGE (value)	Sweep value where the output variable value crosses <i>below</i> a given threshold value

You can define Worst as the highest (HI) or lowest (LO) possible collating function relative to the nominal run.

Procedure

To establish the initial value of the collating function, worst-case analysis begins with a nominal run using all model parameters at their nominal values.

Next, multiple sensitivity analyses determine the individual effect of each model parameter on the collating function. This is accomplished by varying model parameters, one at a time, in consecutive simulations. The

You can define models for nearly all primitive analog circuit parts, such as resistors, capacitors, inductors, and semiconductor devices. PSpice reads the standard model parameter tolerance syntax specified in the .MODEL statement. For each model parameter, PSpice uses the nominal, minimum, and maximum probable values, and the DEV and/or LOT specifiers; the probability distribution type (such as UNIFORM or GAUSS) is ignored.

You can use analog behavioral models to measure waveform characteristics other than those detected by the available collating functions, such as rise time or slope. You can also use analog behavioral models to incorporate several voltages and currents into one output variable to which a collating function may be applied. See [Chapter 6, Analog behavioral modeling](#), for more information.

This procedure saves time by performing the minimum number of simulations required to make an educated guess at the parameter values that produce the worst results. It also has some limitations, which are described in the following sections.

direction (*better* or *worse*) in which the collating function changes with a small increase in each model parameter is recorded.

Finally, for the worst-case run, each parameter value is taken as far from its nominal as allowed by its tolerance, in the direction which should cause the collating function to be its worst (given by the HI or LO specification).

Outputs

A summary of the sensitivity analysis is printed in the PSpice output file (.OUT). This summary shows the percent change in the collating function corresponding to a small change in each model parameter. If a .PROBE statement is included in the circuit file, then the results of the nominal and worst-case runs are saved for viewing in the Probe window.

Caution: An important condition for correct worst-case analysis

Worst-case analysis is not an optimization process; it does not *search* for the set of parameter values that result in the worst result.

It assumes that the worst case occurs when each parameter has been either pushed to one of its limits or left at its nominal value as indicated by the sensitivity analysis. **It shows the true worst-case results when the collating function is *monotonic* within all tolerance combinations.**

Otherwise, there is no guarantee. Usually you cannot be certain whether this condition is true, but insight into the operation of the circuit may alert you to possible anomalies.

Worst-case analysis example

The schematic shown in Figure 84 is for an amplifier circuit that is a biased BJT. This circuit is used to demonstrate how a simple worst-case analysis works. It also shows how *non-monotonic* dependence of the output on a single parameter can adversely affect the worst-case analysis.

Because an AC (small-signal) analysis is being performed, setting the input to unity means that the output, $V_m([OUT])$, is the magnitude of the gain of the amplifier. The only variable declared in this circuit is the resistance of $Rb2$. Because the value of $Rb2$ determines the bias on the BJT, it also affects the amplifier's gain.

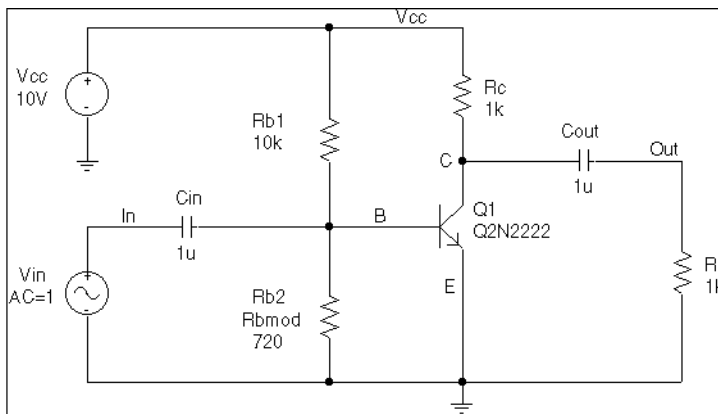


Figure 84 Simple biased BJT amplifier.

Figure 85 is the circuit file used to run one of the following:

- a parametric analysis (.STEP, shown enabled in the circuit file) that sets the value of resistor $Rb2$ by stepping model parameter R through values spanning the specified DEV tolerance range, or
- a worst-case analysis (shown disabled in the circuit file) that allows PSpice to determine the worst-case value for parameter R based upon a sensitivity analysis.

Only one of these analyses can run in any given simulation.

Note *The AC and worst-case analysis specifications (.AC and .WC statements) are written so that the worst-case analysis tries to minimize $V_m([OUT])$ at 100 kHz.*

The netlist and circuit file in Figure 85 are set up to run either a parametric (.STEP) or worst-case (.WC) analysis of the specified AC analysis. These simulations demonstrate the conditions under which worst-case analysis works well and those that can produce misleading results when

```
* Worst-case analysis comparing monotonic and non-monotonic
* output with a variable parameter

.lib

***** Input signal and blocking capacitor *****
Vin In      0      ac      1
Cin In      B      1u

***** "Amplifier" *****
* gain increases with small increase in Rb2, but
* device saturates if Rb2 is maximized.
Vcc Vcc      0      10
Rc  Vcc      C      1k
Q1  C        B      0      Q2N2222
Rb1 Vcc      B      10k
Rb2 B        0      Rbmod 720
.model Rbmod res(R=1 dev 5%)      ; WC analysis results
                                   ; are correct
* .model Rbmod res(R=1.1 dev 5%)   ; WC analysis misled
                                   ; by sensitivity

***** Load and blocking capacitor *****
CoutC      Out      1u
Rl  Out      0      1k

* Run with either the .STEP or the .WC, but not both.
* This circuit file is currently set up to run the .STEP
* (.WC is commented out)

**** Parametric Sweep—providing plot of  $V_m([OUT])$  vs. Rb2 ****
.STEP Res Rbmod(R) 0.8 1.2 10m

***** Worst-case analysis *****
* run once for each of the .model definitions stated above)
* WC AC  $V_m([Out])$  min range 99k 101k list output all

.AC Lin 3 90k 110k
.probe
.end
```

Figure 85 *Amplifier netlist and circuit file.*

output is not monotonic with a variable parameter (see Figure 87 and Figure 88)

For demonstration, the parametric analysis is run first, generating the curve shown in Figure 87 and Figure 88. This curve, derived using the YatX goal function shown in Figure 86 illustrates the non-monotonic dependence of gain on *Rb2*.

```
YatX(1, X_value)=y1{1|sfxv(X_value)!1;}
```

Figure 86 *YatX Goal Function.*

To do this yourself, place the goal function definition in a PROBE.GF file in the circuit directory. Then start PSpice, load all of the AC sweeps, set up the X axis for performance analysis, and add the following trace:

```
YatX(Vm([OUT]),100k)
```

Next, the parametric analysis is commented out and the worst-case analysis is enabled. Two runs are made using the two versions of the *Rbmod* .MODEL statement shown in the circuit file. The model parameter, R, is a multiplier which is used to scale the nominal value of any resistor referencing the *Rbmod* model (*Rb2* in this case).

The first .MODEL statement leaves the nominal value of *Rb2* at 720 ohms. The sensitivity analysis increments R by a small amount and checks its effect on Vm([OUT]). This slight increase in R causes an increase in the base bias voltage of the BJT, and increases the amplifier's gain, Vm([OUT]). The worst-case analysis correctly sets R to its minimum value for the lowest possible Vm([OUT]) (see Figure 87).

Note The YatX goal function is used on the simulation results for the parametric sweep (.STEP) defined in Figure 85. The resulting curves are shown in Figure 87 and Figure 88.

Consider a slightly different scenario: R_{b2} is set to 720 ohms so that maximizing it is not enough to saturate the BJT, but R_{b1} is variable also. The true worst case occurs when R_{b2} is maximized and R_{b1} is minimized. Checking their individual effects is not sufficient, even if the circuit were simulated four times with each resistor in turn set to its extreme values.

Output is monotonic within the tolerance range. Sensitivity analysis correctly points to the minimum value.

The second .MODEL statement scales the nominal value of R_{b2} by 1.1 to approximately 800 ohms. The gain still increases with a small increase in R , but a larger increase in R increases the base voltage so much that it drives the BJT into saturation and nearly eliminates the gain. The worst-case analysis is fooled by the sensitivity analysis into assuming that R_{b2} must be minimized to degrade the gain, but maximizing R_{b2} is much worse (see Figure 88). Note that even an optimizer, which checks the local gradients to determine how the parameters should be varied, is fooled by this circuit.

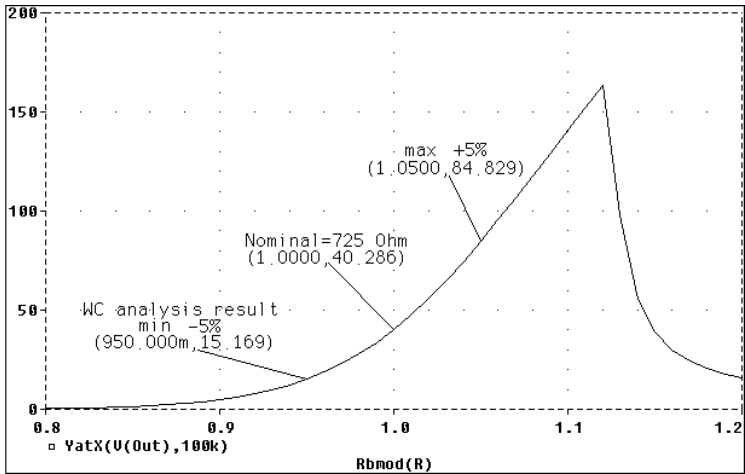


Figure 87 Correct worst-case results.

Output is non-monotonic within the tolerance range, thus producing incorrect worst-case results.

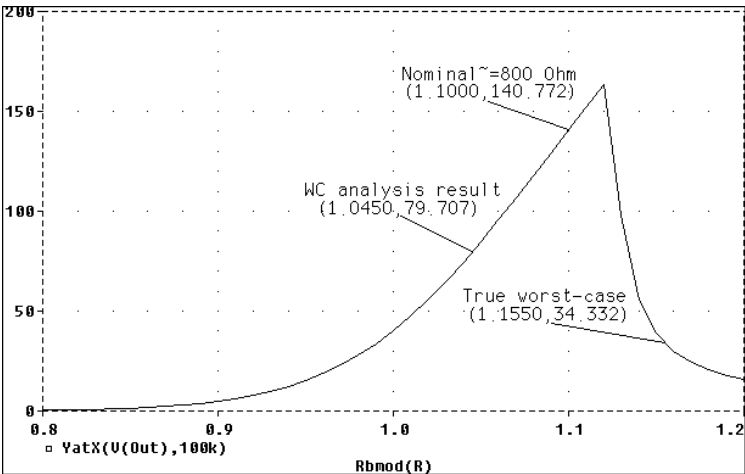


Figure 88 Incorrect worst-case results.

Tips and other useful information

VARY BOTH, VARY DEV, and VARY LOT

When VARY BOTH is specified in the .WC statement and a model parameter is specified with both DEV and LOT tolerances defined, the worst-case analysis may produce unexpected results. The sensitivity of the collating function is only tested with respect to LOT variations of such a parameter.

For example, during the sensitivity analysis, the parameter is varied once affecting all devices referring to it and its effect on the collating function is recorded. For the worst-case analysis, the parameter is changed for all devices by LOT + DEV in the determined direction. See the example schematic in Figure 89 and circuit file in Figure 90.

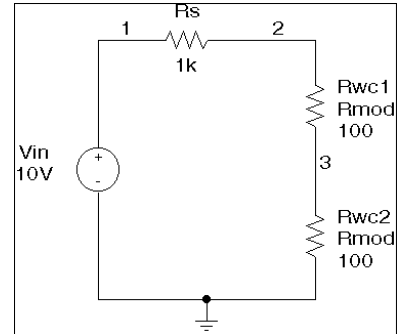


Figure 89 Schematic using VARY BOTH.

```
WCASE  VARY  BOTH    Test
Vin           1      0    10V
Rs           1      2    1K
Rwc1         2      3    Rmod   100
Rwc2         3      0    Rmod   100
.MODEL  Rmod  RES(R=1  LOT 10%  DEV 5%)
.DC Vin      LIST    10
.WC DC      V(3)    MAX    VARY BOTH    LIST    OUTPUT ALL
.ENDS
```

Figure 90 Circuit file using VARY BOTH.

In this case, V(3) is maximized if:

- R_{wc1} and R_{wc2} are both increased by 10% per the LOT tolerance specification, and
- R_{wc1} is decreased by 5% and R_{wc2} is increased by 5% per the DEV tolerance specification.

The final values for R_{wc1} and R_{wc2} should be 105 and 115, respectively. However, because R_{wc1} and R_{wc2} are varied together during the sensitivity analysis, it is assumed that both must be increased to their maximum for a maximum V(3). Therefore, both are increased by 15%.

The purpose of the technique is to reduce the number of simulations. For a more accurate worst-case analysis, you should first perform a worst-case analysis with VARY LOT, manually adjust the nominal model parameter values according to the results, then perform another analysis with VARY DEV specified.

Gaussian distributions

Parameters using Gaussian distributions are changed by 3σ (three times sigma) for the worst-case analysis.

YMAX collating function

This may result in maximizing or minimizing the output variable value over the entire range of the sweep. This collating function is useful when you know the direction in which the maximum deviation occurs.

The purpose of the YMAX collating function is often misunderstood. This function does not try to maximize the deviation of the output variable value from nominal. Depending on whether HI or LO is specified, it tries to maximize or minimize the output variable value itself at the point where maximum deviation occurred during sensitivity analysis.

RELTOL

During the sensitivity analysis, each parameter is varied (multiplied) by $1 + \text{RELTOL}$ where RELTOL is specified in a .OPTIONS statement, or defaults to 0.001.

Sensitivity analysis

The sensitivity analysis results are printed in the output file (.OUT). For each varied parameter, the percent change in the collating function and the sweep variable value at which the collating function was measured are given. The parameters are listed in *worst output* order; for example, the collating function was its worst when the first parameter printed in the list was varied.

When you use the YMAX collating function, the output file also lists mean deviation and sigma values. These are based on the changes in the output variable from nominal at every sweep point in every sensitivity run.

Manual optimization

You can use worst-case analysis to perform *manual optimization* with PSpice. The monotonicity condition is usually met if the parameters have a very limited range.

Performing worst-case analysis with tight tolerances on the parameters produces sensitivity and worst-case results (in the output file). You can use these to decide how the parameters should be varied to achieve the desired response. You can then make adjustments to the nominal values in the circuit file, and perform the worst-case analysis again for a new set of gradients.

Parametric sweeps (.STEP), like the one performed in the circuit file shown in Figure 85, can be used to augment this procedure.

Monte Carlo analysis

Monte Carlo (.MC) analysis may be helpful when worst-case analysis cannot be used. Monte Carlo analysis can often be used to verify or improve on worst-case analysis results. Monte Carlo analysis randomly selects possible parameter values, which can be thought of as randomly selecting points in the *parameter space*. The worst-case analysis assumes that the worst results occur somewhere on the surface of this space, where parameters (to which the output is sensitive) are at one of their extreme values.

If this is not true, the Monte Carlo analysis may find a point at which the results are worse. To try this, replace .WC in the circuit file with .MC <#runs>, where <#runs> is the number of simulations you want to perform. More runs provide higher confidence results. The Monte Carlo summary in the output file lists the runs in decreasing order of collating function value.

To save disk space, do not specify any OUTPUT options.

Next, add the following option to the .MC statement, and simulate again.

```
OUTPUT LIST RUNS <worst_run#>
```

This performs only two simulations: the nominal and the worst Monte Carlo run. The parameter values used during the worst run are written to the output file, and the results of both simulations are saved.

Using Monte Carlo analysis with YMAX is a good way to obtain a conservative guess at the maximum possible deviation from nominal, since worst-case analysis usually cannot provide this information.

Part four

Viewing results

Part four describes the ways to view simulation results.

- [**Chapter 13, Analyzing waveforms**](#), describes how to perform graphical waveform analysis of simulation results.
- [**Chapter 14, Other output options**](#), describes the special symbols you can place on your schematic to generate additional information to the PSpice output file and PSpice window.

Analyzing waveforms

13

Chapter overview

This chapter describes how to perform graphical waveform analysis of simulation results in PSpice. This chapter includes the following:

- [Overview of waveform analysis on page 13-320](#)
- [Setting up waveform analysis on page 13-324](#)
- [Viewing waveforms on page 13-327](#)
- [Analog example on page 13-341](#)
- [User interface features for waveform analysis on page 13-344](#)
- [User interface features for waveform analysis on page 13-344](#)
- [Tracking simulation messages on page 13-354](#)
- [Trace expressions on page 13-356](#)

Overview of waveform analysis

You can use the waveform analysis features of PSpice to visually analyze and interactively manipulate the waveform data produced by circuit simulation.

PSpice uses high-resolution graphics so you can view the results of a simulation both on the screen and in printed form. On the screen, waveforms appear as plots displayed in Probe windows within the PSpice workspace.

In effect, waveform analysis is a software oscilloscope. Performing a PSpice simulation corresponds to building or changing a breadboard, and performing waveform analysis corresponds to looking at the breadboard with an oscilloscope.

With waveform analysis you can:

- View simulation results in multiple Probe windows
- Compare simulation results from multiple circuit designs in a single Probe window
- Display simple voltages, currents, and noise data
- Display complex arithmetic expressions that use the basic measurements
- Display Fourier transforms of voltages and currents, or of arithmetic expressions involving voltages and currents
- Add text labels and other annotation symbols for clarification

PSpice generates two forms of output: the simulation output file and the waveform data file. The calculations and results reported in the simulation output file act as an audit trail of the simulation. However, the graphical analysis of information in the waveform data file is the most informative and flexible method for evaluating simulation results.

Elements of a plot

A single plot consists of the analog (lower) area and the digital (upper) area.

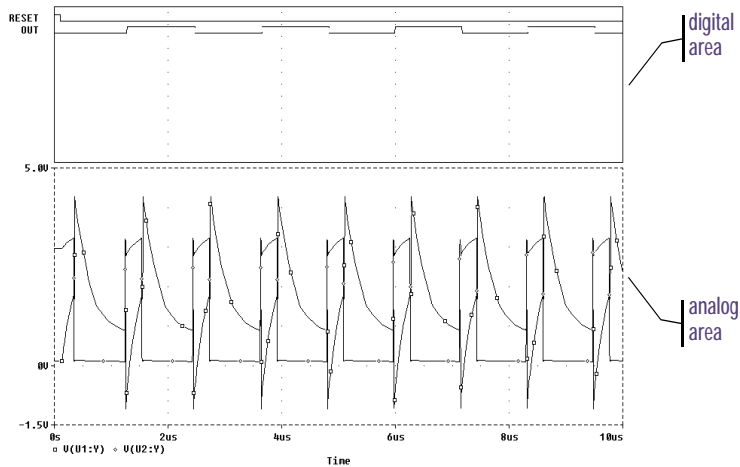


Figure 91 *Analog and digital areas of a plot.*

You can display multiple plots at a time. If you display only analog waveforms, the entire plot will be an analog area. Likewise, if you display only digital waveforms, the entire plot will be a digital area.

Elements of a Probe window

A Probe window is a separately managed waveform display area. A Probe window can include multiple analog and digital plots. Figure 92 shows two plots displayed together.

Because a Probe window is a window object, you can minimize and maximize windows, or move and scale the windows, within the PSpice workspace. A toolbar can be displayed in the Probe window and applies to the active Probe window.

From the View menu, choose Toolbar to display or hide the toolbar.

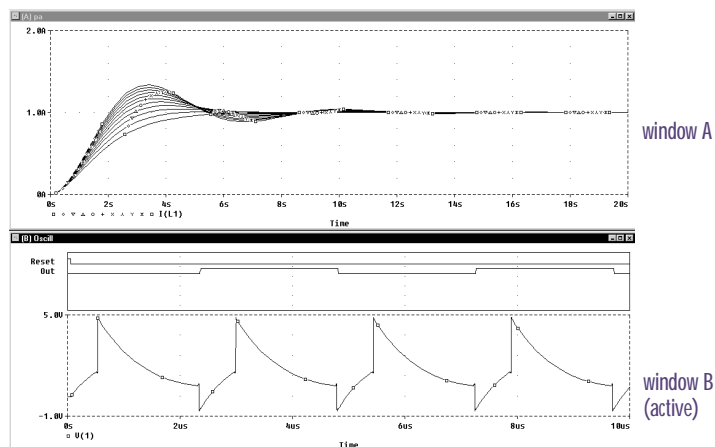


Figure 92 Two Probe windows.

You can display information from one or more waveform data files in one Probe window. After the first file is loaded, load other files into the same Probe window by appending them in PSpice.

Managing multiple Probe windows

You can open any number of Probe windows. Each Probe window is a tab on the worksheet displayed in the middle of the workspace.

The same waveform data file can be displayed in more than one Probe window. You can tile the windows to compare data.

Only one Probe window is active at any given time, identified by a highlighted title bar or a topmost tab. Menu, keyboard, and mouse operations affect only the active Probe window. You can switch to another Probe window by clicking another tab or title bar.

Printing multiple windows

You can print all or selected Probe windows, with up to nine windows on a single page. When you choose Print from the File menu, a list of all open Probe windows appears. Each Probe window is identified by the unique identifier in parentheses in its title bar.

The arrangement of Probe windows on the page can be customized using the Page Setup dialog box. You can print in either portrait (vertical) or landscape (horizontal) orientation. You can also use Print Preview to view all of the Probe windows as they will appear when printed.

Setting up waveform analysis

Setting up colors

You can configure Probe display and print colors in:

- The configuration file, PSPICE.INI
- The Probe Options dialog box

For information on how to use the available colors and color order in a Probe window, see [Configuring trace color schemes on page 13-326](#).

Editing display and print colors in the PSPICE.INI file

In the PSPICE.INI file, you can control the following print and display color settings for Probe windows:

- The colors used to display traces
- The colors used for the Probe window foreground and background
- The order colors are used to display traces
- The number of colors used to display traces

Colors for all items are specified as `<item name>=<color>`. The item names and what they represent are listed in Table 1.

Here are the color names you can specify:

brightcyan	brightblue
brightgreen	brightred
brightmagenta	
brightwhite	
brightyellow	darkblue
darkcyan	darkgray
darkmagenta	darkgreen
darkpink	darkred
lightgreen	
lightblue	lightgray
magenta	mustard
pink	purple
brown	blue
white	black

To edit display and print colors in the PSPICE.INI file

Note After editing PSPICE.INI, you must restart PSpice before your changes will take effect.

- 1 In a standard text editor (such as Notepad), open PSPICE.INI. (This file is normally located in the C:\Windows directory.)
- 2 Scroll to the [PROBE DISPLAY COLORS] or [PROBE PRINTER COLORS] section of the file.
- 3 Add or modify a color entry. See [Table 1 on page 13-325](#) for a description of color entries and their default values. Valid item names include:
 - BACKGROUND
 - FOREGROUND

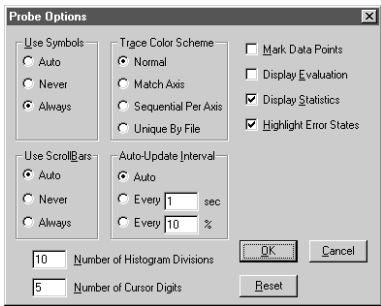
- TRACE_1 through TRACE_12
- 4 If you added or deleted trace number entries, set NUMTRACECOLORS=n to the new number of traces, where n is between 1 and 12. This item represents the number of trace colors displayed on the screen or printed before the color order repeats.
 - 5 Save the file.

Table 1 *Default waveform viewing colors.*

Item Name	Description	Default
BACKGROUND	specifies the color of window background	BLACK
FOREGROUND	specifies the default color for items not explicitly specified	WHITE
TRACE_1	specifies the first color used for trace display	BRIGHTGREEN
TRACE_2	specifies the second color used for trace display	BRIGHTRED
TRACE_3	specifies the third color used for trace display	BRIGHTBLUE
TRACE_4	specifies the fourth color used for trace display	BRIGHTYELLOW
TRACE_5	specifies the fifth color used for trace display	BRIGHTMAGENTA
TRACE_6	specifies the sixth color used for trace display	BRIGHTCYAN

When you want to copy Probe plots to the clipboard and then paste them into a black and white document, choose the Change All Colors to Black option under Foreground in the Copy to Clipboard—Color Filter dialog box (from the Window menu, choose Copy to Clipboard).

For information on what the default available colors and color order are and how to change them, see [Editing display and print colors in the PSPICE.INI file on page 13-324](#).



PSPice saves the selected color scheme for future waveform analyses.

Configuring trace color schemes

In the Probe Options dialog box, you can set options for how the available colors and the color order specified in the PSPICE.INI file are used to display the traces in a Probe window. You can use:

- a different color for each trace
- the same color for all the traces that belong to the same y-axis
- the available colors in sequence for each y-axis
- the same color for all the traces that belong to the same waveform data file

To configure trace color schemes in the Probe Options dialog box

- 1 From the Tools menu, choose Options to display the Probe Options dialog box.
- 2 Under Trace Color Scheme, choose one of the following options:

Table 2

Choose this option...	To do this...
Normal	Use a different color for each trace (for up to 12 traces, depending on the number of colors set in the PSPICE.INI file).
Match Axis	Use the same color for all the traces that belong to the same y-axis. The title of the axis (by default, 1, 2, etc.) is the same color as its traces.
Sequential Per Axis	Use the available colors in sequence for each y-axis.
Unique by File	Use the same color for all the traces in one Probe window that belong to the same waveform data file.

- 3 Click OK.

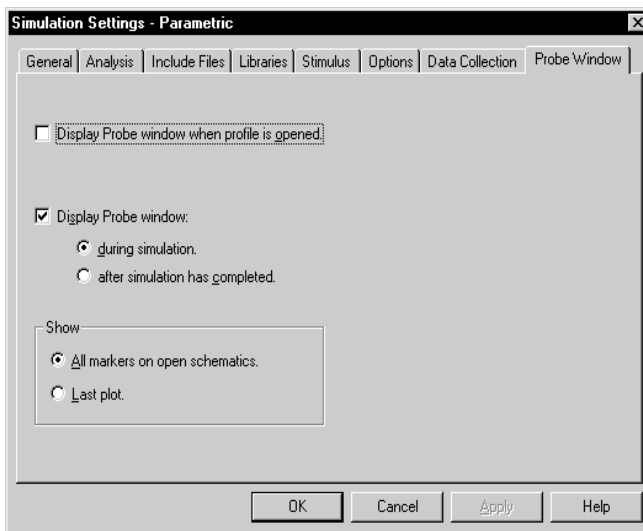
Viewing waveforms

If you are using Capture, you can either view waveforms automatically after you run a simulation, or you can monitor the progress of the simulation as it is running.

Setting up waveform display from Capture

You can configure the way you want to view the waveforms in PSpice by defining display settings in the Probe Window tab in the Simulation Settings dialog box.

You do not need to exit PSpice if you are finished examining the simulation results for one circuit and want to begin a new simulation from within Capture. However, PSpice unloads the old waveform data file for a circuit each time that you run a new simulation of the circuit. After the simulation is complete, the new or updated waveform data file is loaded for viewing.



The display settings in the Probe Window tab are explained in the following table.

Table 3

This setting...	Enables this type of waveform display...
Display Probe window when profile is opened.	Waveforms are displayed only when a .DAT file is opened from within PSpice.
Display Probe window... during simulation.	Waveforms are displayed as the simulation progresses (“marching waveforms”).
Display Probe window... after simulation has completed.	Waveforms are displayed only after the full simulation has completed and all data has been calculated.
Show... all markers on open schematics.	Waveforms are displayed for those nets that have markers attached in the schematic.
Show... last plot.	Waveforms are displayed according to the last display configuration that was used in the Probe window.

Viewing waveforms while simulating

While a simulation is in progress, you can monitor the results for the data section being written by PSpice. This function is only available when the Display Probe window during simulation option is enabled in the Probe Window tab of the Simulation Settings dialog box.

To monitor results during a simulation

- 1 From Capture’s PSpice menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.
- 2 Click the Probe Window tab.
- 3 Select Display Probe window and then click during simulation.
- 4 Click OK to close the Simulation Settings dialog box.

If you open a new Probe window (from the Window menu, choose New Window) while monitoring the data, the new window also starts in monitor mode because it is associated with the same waveform data file.

- 5 From the PSpice menu, choose Run to start the simulation.
One Probe window is displayed in monitor mode.
- 6 Do one of the following to select the waveforms to be monitored:
 - From PSpice's Trace menu, choose Add, and enter one or more trace expressions.
 - From Capture's PSpice menu, point to Markers, then choose and place one or more markers.

The Probe window monitors the waveforms for as long as the most recent data section is being written. After that data section is finished, the window changes to manual mode. To see the full set of runs, you must update the display by using the Add Trace command under the Trace menu.

During a multi-run simulation (such as Monte Carlo, parametric, or temperature), PSpice displays only the data for the most recent run in the Probe window.



or press **Insert**

For more information, see [Using schematic page markers to add](#)
10-001

Configuring update intervals

You can define the frequency at which PSpice updates the waveform display as follows:

- At fixed time intervals (every n sec)
- According to the percentage of simulation completed (every n %), where n is user-defined

The default setting (Auto) updates traces each time PSpice gets new data from a simulation.

To change the update interval

- 1 From the Tools menu, choose Options.
- 2 In the Auto-Update Interval frame, choose the interval type (sec or %), then type the interval in the text box.



Interacting with waveform analysis during simulation

The functions that change the x-axis domain (that set a new x-axis variable) can not be accessed while the simulation is running. If you have enabled the display of

waveforms during simulation and wish to reconfigure the x-axis settings (as explained below), you must wait until the simulation run has finished.

The following table shows how to enable the functions that change the x-axis domain.

Table 4

	Enable this function...	By doing this...
	Fast Fourier transforms	<div><div>1</div><div>From the Plot menu, choose Axis Settings.</div><div>2</div><div>In the Processing Options frame, select Fourier.</div></div>
	Performance analysis	<div><div>1</div><div>From the Plot menu, choose Axis Settings.</div><div>2</div><div>In the Processing Options frame, select Performance Analysis.</div></div>
	New x-axis variable	<div><div>1</div><div>From the Plot menu, choose Axis Settings, then click the X Axis tab.</div><div>2</div><div>Click the Axis Variable button.</div><div>3</div><div>In the X Axis Variable dialog box, specify a new x-axis variable.</div></div>
	Goal function evaluation	<div><div>1</div><div>From the Trace menu, select Eval Goal Function.</div><div>2</div><div>In the Evaluate Goal Function(s) dialog box, specify a goal function.</div></div>
	Load a completed data section	<div><div>1</div><div>From the File menu, choose Append Waveform (.DAT).</div><div>2</div><div>Select a .DAT file to append.</div></div>

Pausing a simulation and viewing waveforms

You can pause a simulation to analyze waveforms before the simulation is finished. After you pause the simulation, you can either resume the simulation or end it.

To pause a simulation

- 1

From PSpice’s Simulation menu, choose Pause.
- 2

In the Probe window, view the waveforms generated before you paused the simulation.

- 3 Do one of the following:
 - From the Simulation menu, choose Run to resume the simulation.
 - From the Simulation menu, choose Stop to stop the simulation.

Using schematic page markers to add traces

You can place markers on a schematic page to identify the points where you want to see waveform results displayed. You can place markers:

See [Trace expressions on page 13-356](#) for ways to add traces within PSpice.

- Before simulation, to limit results written to the waveform data file and automatically display those traces in PSpice.
- During or after simulation, with PSpice running, to automatically display traces in the active Probe window.

The color of the marker you place is the same as its corresponding waveform analysis trace. If you change the color of the trace, the color of the marker on the schematic page changes accordingly.

The Markers submenu also provides options for controlling the display of marked results in PSpice, after initial marker placement, and during or after simulation.

To place markers on a schematic page

- 1 From Capture's PSpice menu, point to Markers, then choose the marker type you want to place. (Some of the markers are from the Advanced submenu.)



Table 5

Waveform	Markers menu command	Advanced submenu command
voltage	Voltage Level	not required
voltage differential	Voltage Differential	not required
current	Current Into Pin	not required
digital signal	Voltage Level	not required
dB*	Advanced	db Magnitude of Voltage db Magnitude of Current
phase*	Advanced	Phase of Voltage Phase of Current
group delay*	Advanced	Group Delay of Voltage Group Delay of Current
real*	Advanced	Real Part of Voltage Real Part of Current
imaginary*	Advanced	Imaginary Part of Voltage Imaginary Part of Current

* You can use these markers instead of the built-in functions provided in output variable expressions (see [Table 12 on page 13-364](#)). However, these markers are only available after defining a simulation profile for an AC Sweep/Noise analysis.

The color of the marker is the same as its corresponding waveform analysis trace. If you change the color of the trace, the color of the marker changes accordingly.

- 2 Point to the wires or pins you wish to mark and click to place the chosen markers.
- 3 Right-click and select End Mode to stop placing markers.
- 4 If you have not simulated the circuit yet, from the PSpice menu, choose Run.

To hide or delete marked results

- 1 From Capture’s PSpice menu, point to Markers, then choose one of the following:

Table 6

Choose this option...	To do this...
Hide All	Hide traces in the waveform analysis display for all markers placed on any page or level of the schematic.
Delete All	Remove all markers from the schematic and all corresponding traces from the waveform analysis display.

Limiting waveform data file size

When PSpice performs a simulation, it creates a waveform data file. The size of this file for a transient analysis is roughly equal to:

$(\# \text{ transistors}) \cdot (\# \text{ simulation time points}) \cdot 24 \text{ bytes}$

The size for other analysis types is about 2.5 times smaller. For long runs, especially transient runs, this can generate waveform data files that are several megabytes in size. Even if this does not cause a problem with disk space, large waveform data files take longer to read in and take longer to display traces on the screen.

You can limit waveform data file size by:

- placing markers on your schematic before simulation and having PSpice restrict the saved data to these markers only
- excluding data for internal subcircuits
- suppressing simulation output

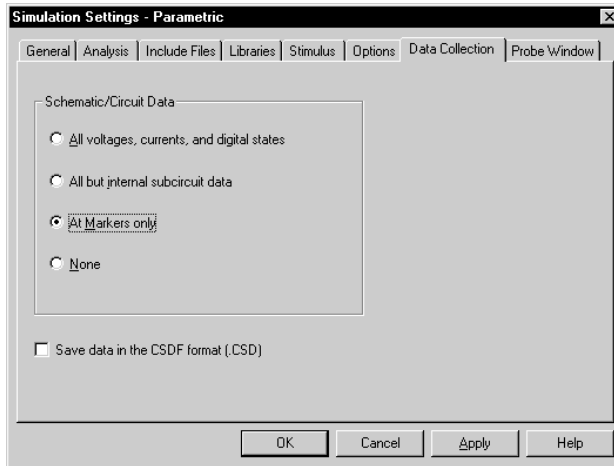
Limiting file size using markers

One reason that waveform data files are large is that, by default, PSpice stores *all net voltages and device currents* for each step (for example, time or frequency points).

However, if you have placed markers on your schematic prior to simulation, PSpice saves only the results for the marked wires and pins.

To limit file size using markers

- 1 From Capture's PSpice menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.



- 2 Click the Data Collection tab.
- 3 In the Schematic/Circuit Data frame, choose At Markers only and click OK.
- 4 From the PSpice menu, point to Markers, then choose the marker type you want to place.
- 5 Point to the wires or pins you wish to mark and click to place the chosen markers.
- 6 Right-click and select End Mode to stop placing markers.
- 7 From the PSpice menu, choose Run to start the simulation.

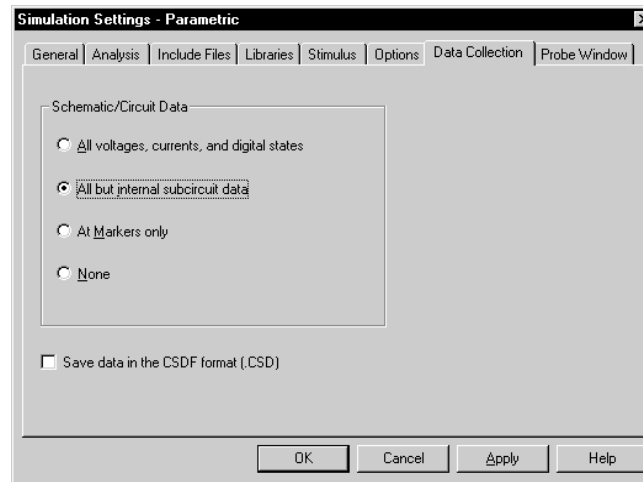
The color of the marker on the schematic page is the same as its corresponding waveform analysis trace. If you change the color of the trace, the color of the marker changes accordingly.

Limiting file size by excluding internal subcircuit data

By default, PSpice saves data for all internal nodes and devices in subcircuit models in a design. You can exclude data for internal subcircuit nodes and devices.

To limit file size by excluding data for internal subcircuits

- 1 From PSpice's Simulation menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.



- 2 Click the Data Collection tab.
- 3 In the Schematic/Circuit Data frame, choose All but internal subcircuit data, then click OK.
- 4 From the PSpice menu, choose Run to start the simulation.

Limiting file size by suppressing the first part of simulation output

Long transient simulations create large waveform data files because PSpice stores many data points. You can suppress a part of the data from a transient run by setting the simulation analysis to start the output at a time later than 0. This does not affect the transient calculations

Suppressing part of the data run also limits the size of the PSpice output file.

themselves—these always start at time 0. This delay only suppresses the output for the first part of the simulation.

To limit file size by suppressing the first part of transient simulation output

- 1 From Capture's PSpice menu, choose Edit Simulation Settings to display the Simulation Settings dialog box.
- 2 Click the Analysis tab.
- 3 From the Analysis type list, select the Time Domain (Transient) option.
- 4 In the Start saving data after text box, type a delay time.
- 5 Click OK to close the Simulation Settings dialog box.
- 6 From the PSpice menu, choose Run to start the simulation.

The simulation begins, but no data is stored until after the delay has elapsed.

Using simulation data from multiple files

You can load simulation data from multiple files into the same Probe window by appending waveform data files.

When more than one waveform data file is loaded, you can add traces using all loaded data, data from only one file, or individual data sections from one or more files.

Appending waveform data files

To append a waveform data file

- 1 In PSpice, from the File menu, choose Append Waveform (.DAT).
- 2 Select a *.DAT file to append, and click OK.



- 3 If the file has multiple sections of data for the selected analysis type, the Available Sections dialog box appears. Do one of the following:
 - Click the sections you want to use.
 - Click the All button to use all sections.
- 4 Click OK.

Adding traces from specific loaded waveform data files

If two or more waveform data files have identical simulation output variables, trace expressions that include those variables generate traces for each file. However, you can specify which waveform data file to use in the trace expression. You can also determine which waveform data file was used to generate a specific trace.

To add a trace from a specific loaded waveform data file

- 1 In PSpice, from the Trace menu, choose Add Trace to display the Add Traces dialog box.
- 2 In the Trace Expression text box, type an expression using the following syntax:

trace_expression@fn

where *n* is the numerical order (from left to right) of the waveform data file as it appears in the PSpice title bar, or

trace_expression@s@fn

where *s* is a specific data section of a specific waveform data file.

- 3 Click OK.

To identify the source file for an individual trace

- 1 In the trace legend, double-click the symbol for the trace you want to identify (Figure 93).

The Section Information dialog box appears, containing the trace name and—if there is more than



The Simulation Output Variables list in the Add Traces dialog box contains the output variables for all loaded waveform data files.

Example: To plot the V(1) output for data section 1 from the second data file loaded, type the following trace expression:

`V(1)@1@f2`

You can also use the name of the loaded data file to specify it. For example, to plot the V(1) output for all data sections of a loaded data file, MYFILE.DAT, type the following trace expression:

`V(1)@"MYFILE.DAT"`

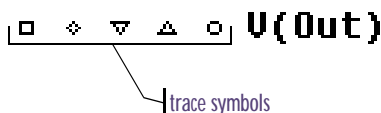


Figure 93 Trace legend symbols.

one waveform data file loaded in the plot—the full path for the file from which the trace was generated.

Also listed is information about the simulation that generated the waveform data file and the number of data points used (Figure 94).

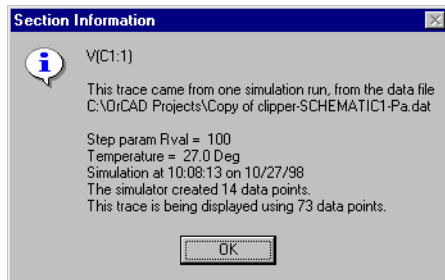


Figure 94 *Section information message box.*

Saving simulation results in ASCII format

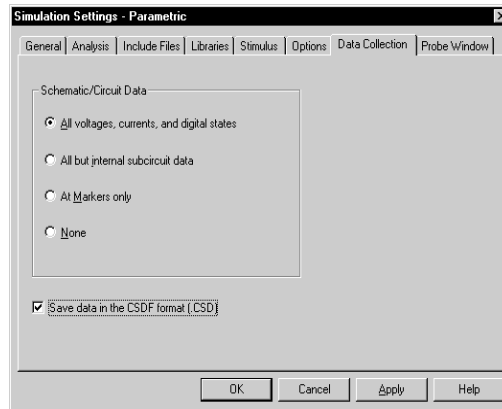
The default waveform data file format is binary. However, you can save the waveform data file in the Common Simulation Data Format (CSDF) instead.

Warning: Data files saved in the CSDF format are two or more times the size of binary files.

When you first open a CSDF data file, PSpice converts it back to the .DAT format. This conversion takes two or more times as long as opening a .DAT file. PSpice saves the new .DAT file for future use.

To save simulation results in ASCII format

- 1 From PSpice's Simulation menu, choose Edit Profile to display the Simulation Settings dialog box.



- 2 Click the Data Collection tab.
- 3 Select Save data in the CSDF format (.CSD).
- 4 Click OK.

PSpice writes simulation results to the waveform data file in ASCII format (as *.CSD instead of *.DAT), following the CSDF convention.

Analog example

In this section, basic techniques for performing waveform analysis are demonstrated using the analog circuit EXAMPLE.OPJ.

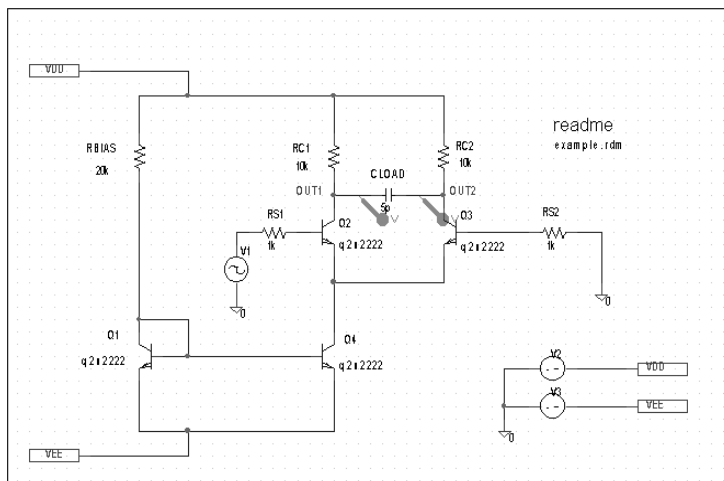


Figure 95 Example schematic EXAMPLE.OPJ.

Running the simulation

The simulation is run with the Bias Point Detail, Temperature, and Transient analyses enabled. The temperature analysis is set to 35 degrees. The transient analysis is setup as follows:

Print Step	20ns
Final Time	1000ns
Enable Fourier	selected
Center Frequency	1Meg
Output Vars	V(OUT2)

To start the simulation

- 1 From Capture's File menu, point to Open and choose Project.
- 2 Open the following project in your OrCAD program installation directory:

The example project EXAMPLE.OPJ is provided with your OrCAD programs.

When shipped, EXAMPLE.OPJ is set up with multiple analyses. For this example, the AC sweep, DC sweep, Monte Carlo/worst-case, and small-signal transfer function analyses have been disabled. The specification for each of these disabled analyses remains intact. To run them from Capture in the future, from the PSpice menu, choose Edit Simulation Settings and enable the analyses.

Note When you run a Fourier analysis using PSpice as specified in this example, PSpice writes the results to the PSpice output file (*.OUT). You can also use Probe windows to display the Fourier transform of any trace expression by using the FFT capability in PSpice. To find out more, refer to PSpice A/D online Help.

\PSPICE\SAMPLES\ANASIM\EXAMPLE\EXAMPLE.OPJ

If PSpice is set to show traces for all markers on startup, you will see the V(OUT1) and V(OUT2) traces when the Probe window displays. To clear these traces from the plot, from the Trace menu, choose Delete All Traces.

- 3 From the PSpice menu, choose Run to start the simulation.

PSpice generates a binary waveform data file containing the results of the simulation. A new Probe window appears with the waveform data file EXAMPLE.DAT already loaded (Figure 96).

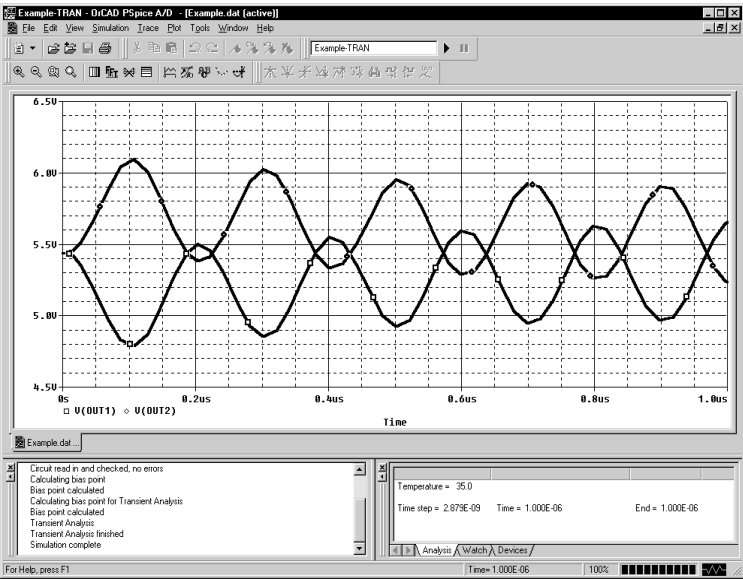


Figure 96 Waveform display for EXAMPLE.DAT.

Because this sample project was set up as a transient analysis type, the data currently loaded are the results of the transient analysis.

Note In this sample, the voltage markers for OUT1 and OUT2 are already placed in the design. If the markers are not placed prior to simulating, you can display the waveforms later, as explained below in Displaying voltages on nets.

Displaying voltages on nets

After selected an analysis, voltages on nets and currents into device pins can be displayed in the Probe windows using either schematic markers or output variables (as will be demonstrated in this example).

To display the voltages at the OUT1 and OUT2 nets using output variables

- 1 From the Trace menu, choose Add Trace to display the Add Traces dialog box.

press Insert

The Simulation Output Variables frame displays a list of valid output variables.

- 2 Click V(OUT1) and V(OUT2), then click OK. The Probe window should look similar to Figure 96.

User interface features for waveform analysis

PSpice provides direct manipulation techniques and shortcuts for analyzing waveform data. These techniques are described below.

Shortcut keys

Many of the menu commands in PSpice have equivalent keyboard shortcuts. For instance, after placing a selection rectangle in the analog portion of the plot, you can type **Ctrl+A** instead of choosing Area from the View menu. For a list of shortcut keys, search on Keyboard Shortcuts in PSpice Help.



Click anywhere on the plot to remove the selection rectangle without zooming.

Zoom regions

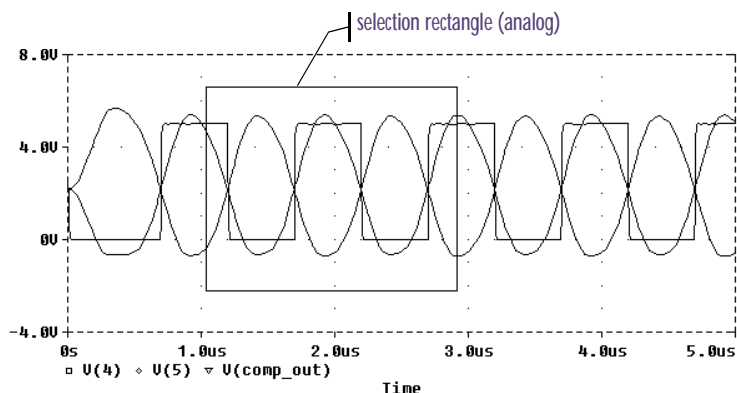
PSpice provides a direct manipulation method for marking the zoom region in the analog area of the plot.

To zoom in or out

- 1 Do one of the following on the toolbar:
 - Click the View In toolbar button to zoom in by a factor of 2 around the point you specify.
 - Click the View Out toolbar button to zoom out by a factor of 2 around the point you specify.

To zoom in the analog area using the mouse

- 1 Drag the mouse pointer to make a selection rectangle as shown below.



- 2 From the View menu, point to Zoom, then choose Area.



PSpice changes the plot to display the region within the selection rectangle.

Scrolling traces

By default, when a plot is zoomed, standard scroll bars appear to the right or at the bottom of the plot area as necessary. These can be used to pan through the data. You can configure scroll bars so they are always present or are never displayed.

To configure scroll bars

- 1 In PSpice, from the Tools menu, choose Options.
- 2 In the Use Scroll Bars frame, choose one of the scroll bars options, as described below.

Table 1

Choose this option...	To do this...
Auto	Have scroll bars appear when a plot is zoomed or when additional traces are displayed in the plot but are not visible (default).
Never	Never display scroll bars. This mode provides maximum plot size and is useful on VGA and other low resolution displays.
Always	Display scroll bars at all times. However, they are disabled if the corresponding axis is full scale.

Modifying trace expressions and labels

For information about adding labels (including text, line, poly-line, arrow, box, circle, ellipse, and mark), refer to the online Help in PSpice.

You can modify trace expressions, text labels, and ellipse labels that are currently displayed within the Probe window, thus eliminating the need to delete and recreate any of these objects.

To modify trace expressions

- 1 Click the trace name to select it (selection is indicated by a color change).
- 2 From the Edit menu, choose Modify Object.
- 3 In the Modify Trace dialog box, edit the trace expression just as you would when adding a trace.

You can also double-click the trace name to modify the trace expression.

For more information on adding traces, see [Adding traces from specific loaded waveform data files on page 13-338](#) and [To add traces using output variables on page 13-356](#).

To modify text and ellipse labels

- 1 Click the text or ellipse to select it (selection is indicated by a color change).
- 2 From the Edit menu, choose Modify Object.
- 3 Edit the label by doing one of the following:
 - In the Ellipse Label dialog box, change the inclination angle.
 - In the Text Label dialog box, change the text label.

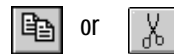
You can also double-click a text or ellipse label to modify it.

Moving and copying trace names and expressions

Trace names and expressions can be selected and moved or copied, either within the same Probe window or to another Probe window.

To copy or move trace names and expressions

- 1 Click one or more (**Shift**+click) trace names. Selected trace names are highlighted.
- 2 From the Edit menu, choose Copy or Cut to save the trace names and expressions to the clipboard. Cut removes trace names and traces from the Probe window.
- 3 In the Probe window where traces are to be added, do one of the following:
 - To add trace names to the end of the currently displayed set, choose Paste from the Edit menu.



or



or press **Ctrl**+**V**

When adding a trace to a Probe window, you can make the trace display name different from the trace expression:

- 1 From the Trace menu, choose Add Trace.
- 2 In the Trace Expression text box, enter a trace expression using the syntax:
`trace_expression[:display_name]`
- 3 Click OK.

- To add traces before a currently displayed trace name, select the trace name and then choose Paste from the Edit menu.

Here are some considerations when copying or moving trace names and expressions into a different Probe window:

- If the new Probe window is reading the same waveform data file, the copied or moved trace names and expressions display traces that are identical to the original selection set.
- If the new Probe window is reading a *different* waveform data file, the copied or moved names and expressions display different traces generated from the new data.

For example, suppose two waveform data files, MYSIM.DAT and YOURSIM.DAT each contain a V(2) waveform. Suppose also that two Probe windows are currently displayed where window A is loaded with MYSIM.DAT, and window B is loaded with YOURSIM.DAT.

When V(2) is copied from window A to window B, the trace looks different because it is determined by data from YOURSIM.DAT instead of MYSIM.DAT.

For information about adding labels (including text, line, poly-line, arrow, box, circle, ellipse, and mark), refer to the online Help in PSpice.



Copying and moving labels

Labels can be selected and moved or copied, either within the same Probe window or to another Probe window.

To copy labels

- 1 Select one or more (**Shift**+click) labels, or select multiple labels by drawing a selection rectangle. Selected labels are highlighted.
- 2 From the Edit menu, choose Copy or Cut to save the labels to the clipboard.

Cut removes labels from the Probe window.

- 3 Switch to the Probe window where labels are to be added, and from the Edit menu, choose Paste.
- 4 Click on the new location to place the labels.

press **Ctrl** + **V**

To move labels

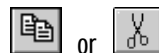
- 1 Select one or more (**Shift**+click) labels, or select multiple labels by drawing a selection rectangle. Selected labels are highlighted.
- 2 Move the labels by dragging them to a new location.

Tabulating trace data values

You can generate a table of data points reflecting one or more traces in the Probe window and use this information in a document or spreadsheet.

To view the trace data values table

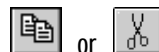
- 1 Select one or more (**Shift**+click) trace names. Selected trace names are highlighted.
- 2 From the Edit menu, choose Copy or Cut to save the trace data point values to the Clipboard.
Cut removes traces from the Probe window.
- 3 In Clipboard Viewer, from the Display menu, choose either Text or OEM Text.



To export the data points to other applications

- 1 Select one or more (**Shift**+click) trace names. Selected trace names are highlighted.
- 2 From the Edit menu, choose Copy or Cut to save the trace data point values to the Clipboard.
Cut removes traces from the Probe window.
- 3 Paste the data from the Clipboard into a text editor, a spreadsheet program, or a technical computing program (such as Mathcad).

Saving the data directly to a file from Clipboard Viewer can create superfluous data at the beginning of the file.



4 Save the file.

Using cursors

When one or more traces are displayed, you can use cursors to display the exact coordinates of two points on the same trace, or points on two different traces. In addition, differences are shown between the corresponding coordinate values for the two cursors.

Displaying cursors

To display both cursors

press **Ctrl** + **⇧ Shift** + **C**

You can move the cursor box anywhere over the Probe window by dragging the box to another location.

- 1 From the Trace menu, point to Cursor, then choose Display.

The Probe Cursor window appears, showing the current position of the cursor on the x-axis and y-axis. As you move the cursors, the values in the cursor box change.

In the analog area of the plot (if any), both cursors are initially placed on the trace listed first in the trace legend. The corresponding trace symbol is outlined with a dashed line.

Moving cursors

To move cursors along a trace using menu commands

- 1 From the Trace menu, point to Cursor, then choose Peak, Trough, Slope, Min, Max, Point, or Search.

For more information about the cursor commands, refer to the online Help in PSpice.

To move cursors along a trace using the mouse

- 1 Use the right and left mouse buttons as described in [Table 2](#) below.

For a family of curves (such as from a nested DC sweep), you can use the mouse or the arrow keys to move the cursor to one of the other curves in the family. You can also click the desired curve.









Table 2 *Mouse actions for cursor control*

Click this...	To do this with the cursors...
cursor assignment	
Left-click the analog trace symbol.	Associate the first cursor with the selected trace.
Right-click the analog trace symbol.	Associate the second cursor with the selected trace.
cursor movement	
Left-click in the display area.	Move the first cursor to the closest trace segment at the X position.
Right-click in the display area.	Move the second cursor to the closest trace segment at the X position.

To move cursors along a trace using the keyboard

- 1 Use key combinations as described in [Table 3](#) below.

Table 3 *Key combinations for cursor control*

Us this key combination...	To do this with the cursors...
 and 	Change the trace associated with the first cursor.
 and 	Change the trace associated with the second cursor.
 and 	Move the first cursor along the trace.
 and 	Move the second cursor along the trace.

To place a label, click Plot, point to Label and then choose the desired type of object you want to place.

Table 3 Key combinations for cursor control (continued)

Us this key combination...	To do this with the cursors...
Home	Move the first cursor to the beginning of the trace.
Shift+Home	Move the second cursor to the beginning of the trace.
End	Move the first cursor to the end of the trace.
Shift+End	Move the second cursor to the end of the trace.

Example: using cursors

Figure 97 shows both cursors on the V(1) waveform in the analog area of the plot.

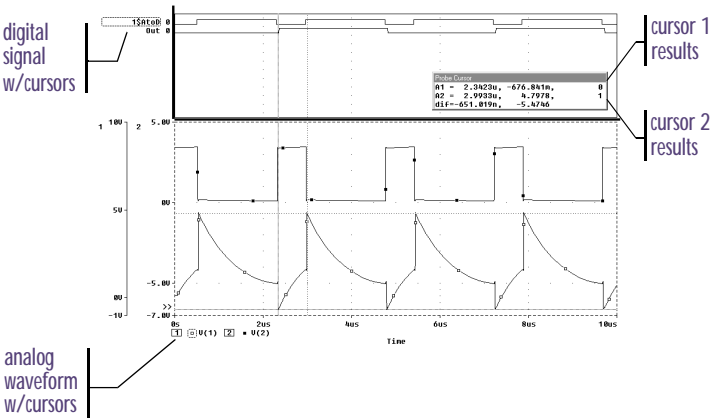


Figure 97 Cursors positioned on a trough and peak of V(1)

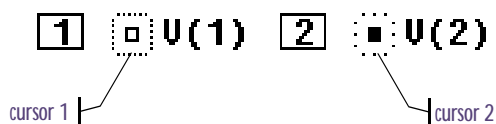
To position a cursor on the next trough of a waveform, from the Trace menu, point to Cursor, then choose Trough.

To position a cursor on the next peak of a waveform, from the Trace menu, point to Cursor, then choose Peak.

For more information about cursors, refer to the online Help in PSpice.

Cursor 1 is positioned on the first trough (dip) of the V(1) waveform. Cursor 2 is positioned on the second peak of the same waveform. In the Probe Cursor window, cursor 1 and cursor 2 coordinates are displayed (A1 and A2, respectively) with their difference shown in the bottom line (dif). The logic state of the Out signal is also displayed to the right of the cursor coordinates.

The mouse buttons are also used to associate each cursor with a different trace by clicking appropriately on either the analog trace symbol in the legend. These are outlined in the pattern corresponding to the associated cursor's crosshair pattern. Given the example in Figure 97, right-clicking the V(2) symbol will associate cursor 2 with the V(2) waveform. The analog legend now appears as shown below.



The Probe Cursor window also updates the A2 coordinates to reflect the X and Y values corresponding to the V(2) waveform.

Tracking simulation messages

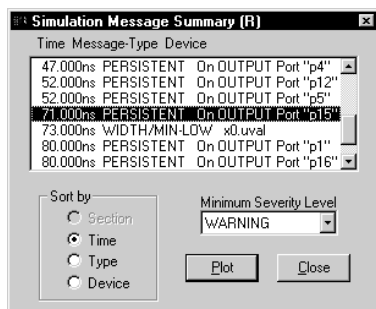
PSpice provides explanatory messages for errors that occur during simulation with their corresponding waveforms. You can view messages from:

- the Simulation Message Summary dialog box, or
- the waveform display.

Message tracking from the message summary

A message summary is available for simulations where diagnostics have been logged to the waveform data file. You can display the message summary:

- When loading a waveform data file (click OK when the Simulation Errors dialog box appears).
- Anytime by choosing Simulation Messages from the View menu.



Example: If you select WARNING as the minimum severity level, the Simulation Message Summary dialog box will display WARNING, SERIOUS, and FATAL messages.

The Simulation Message Summary dialog box

The Simulation Message Summary dialog box lists message header information. You can filter the messages displayed in the list by selecting a severity level from the Minimum Severity Level drop-down menu. Messages are categorized (in decreasing order of severity) as FATAL, SERIOUS, WARNING, or INFO (informational).

When you select a severity level, the Message Summary displays only those messages with the chosen severity *or higher*. By default, the minimum severity level displayed is SERIOUS.

To display waveforms associated with messages

- 1 In the Simulation Message Summary dialog box, double-click a message.

For most message conditions, a Probe window appears that contains the waveforms associated with the simulation condition, along with detailed message text.

Persistent hazards

If a PERSISTENT HAZARD message is displayed, two plots appear (see Figure 98), containing the following:

- the waveforms that initially caused the timing violation or hazard (lower plot)
- the primary outputs or internal state devices to which the condition has propagated (upper plot)

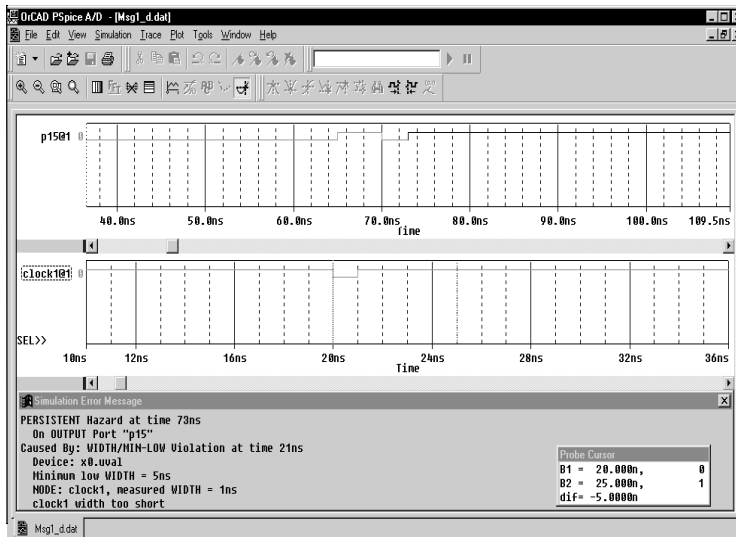


Figure 98 Waveform display for a persistent hazard.

Message tracking from the waveform

Trace segments with associated diagnostics are displayed in the foreground color specified in your PSPICE.INI file. This color is different from those used for standard state transitions.

To display explanatory message text

- 1 Double-click within the tagged region of a trace.

Trace expressions

Traces are referred to by output variable names. Output variables are similar to the PSpice output variables specified in the Simulation Settings dialog box for noise, Monte Carlo, worst-case, transfer function, and Fourier analyses. However, there are additional alias forms that are valid for trace expressions. Both forms are discussed here.

To add traces using output variables

- 1 From the Trace menu, choose Add Trace to display the Add Traces dialog box.
- 2 Construct a trace expression using any combination of these controls:
 - In the Simulation Output Variables frame, click output variables.
 - In the Functions or Macros frame, select operators, functions, constants, or macros.
 - In the Trace Expression text box, type in or edit output variables, operators, functions, constants, or macros.
- 3 If you want to change the name of the trace expression as it displays in the Probe window, use the following syntax:

You can display a subset of the available simulation output variables by selecting or clearing the variable type check boxes in the Simulation Output Variables frame. Variable types not generated by the circuit simulation are dimmed.

For more information about trace expressions, see [Analog trace expressions on page 13-364](#).

trace expression; display name

4 Click OK.

Basic output variable form

This form is representative of those used for specifying some PSpice analyses.

<output>[A C suffix](<name>[,name])

Table 4

This placeholder...	Means this...
<i><output></i>	type of output quantity: V for voltage or I for current
<i>[A C suffix]*</i>	quantity to be reported for an AC analysis. For a list of valid AC suffixes, see Table 8 on page 13-361
<i><name>[,name]</i>	<p>specifies either the <i>net</i> or (+ <i>net</i>, - <i>net</i>) pair for which the voltage is to be reported, or the <i>device</i> for which a current is reported, where:</p> <ul style="list-style-type: none"> <i>net</i> specifies either the <i>net</i> or <i>pin id</i> (<fully qualified device name>:<pin name>) <i>device name</i> specifies the fully qualified device name; for a list of device types, see Table 9 on page 13-361 and Table 10 on page 13-362

A fully qualified device name consists of the full hierarchical path followed by the device's reference designator. For information about syntax, see the voltage output variable naming rules.

Output variable form for device terminals

This form can only be specified for trace expressions. The primary difference between this and the basic form is that the terminal symbol appears before the *net* or *device name* specification (whereas the basic form treats this as the *pin name* within the *pin id*).

`<output>[terminal]*[A C suffix](<name>[,name])`

Table 5

This placeholder...	Means this...
<code><output></code>	type of output quantity: V for voltage, I for current, or N for noise
<code>[terminal]*</code>	one or more terminals for devices with more than two terminals; for a list of terminal IDs, see Table 10 on page 13-362
<code>[A C suffix]*</code>	quantity to be reported for an AC analysis; for a list of valid AC suffixes, see Table 8 on page 13-361
<code><name>[,<name>])</code>	<i>net</i> , <i>net</i> pair, or fully qualified <i>device name</i> ; for a list of device types, see Table 9 on page 13-361 and Table 10 on page 13-362

[Table 6 on page 13-358](#) summarizes the valid output formats. [Table 7 on page 13-360](#) provides examples of equivalent output variables. Note that some of the output variable formats are unique to trace expressions.

Table 6 Output variable formats

Format	Meaning
Voltage variables	
<code>V[ac](< +analog net > [,< -analog net >])</code>	Voltage between + and - <i>analog net ids</i>
<code>V<pin name>[ac](< device >)</code>	Voltage at <i>pin name</i> of a <i>device</i>

Table 6 *Output variable formats (continued)*

Format	Meaning
V< x >[ac](<i>< 3 or 4-terminal device ></i>)	Voltage at non-grounded terminal <i>x</i> of a <i>3 or 4-terminal device</i>
V< z >[ac](<i>< transmission line device ></i>)	Voltage at end <i>z</i> of a <i>transmission line device</i> (<i>z</i> is either <i>A</i> or <i>B</i>)
Current variables	
I[ac](<i>< device ></i>)	Current into a <i>device</i>
I< x >[ac](<i>< 3 or 4-terminal device ></i>)	Current into terminal <i>x</i> of a <i>3 or 4-terminal device</i>
I< z >[ac](<i>< transmission line device ></i>)	Current into end <i>z</i> of a <i>transmission line device</i> (<i>z</i> is either <i>A</i> or <i>B</i>)
Sweep variables	
< <i>DC sweep variable</i> >	name of any variable used in the DC sweep analysis
FREQUENCY	AC analysis sweep variable
TIME	transient analysis sweep variable
Noise variables	
V[db](ONoise)	total RMS-summed noise at output net

Table 6 *Output variable formats (continued)*

Format	Meaning
V[db](INOISE)	total equivalent noise at input source
NTOT(ONOISE)	sum of all noise contributors in the circuit
N< noise type >(< device name >)	contribution from <i>noise type</i> of <i>device name</i> to the total output noise*

* See [Table 11 on page 13-363](#) for a complete list of noise types by device type. For information about noise output variable equations, the units used to represent noise quantities in trace expressions, and a noise analysis example, see [Analyzing Noise in the Probe window on page 9-245](#).

Table 7 *Examples of output variable formats*

A basic form	An alias equivalent	Meaning
V(NET3,NET2)	(same)	voltage between analog nets labeled NET3 and NET2
V(C1:1)	V1(C1)	voltage at pin1 of capacitor C1
VP(Q2:B)	VBP(Q2)	phase of voltage at base of bipolar transistor Q2
V(T32:A)	VA(T32)	voltage at port A of transmission line T32
I(M1:D)	ID(M1)	current through drain of MOSFET device M1
VIN	(same)	voltage source named VIN
FREQUENCY	(same)	AC analysis sweep variable
NFID(M1)	(same)	flicker noise from MOSFET M1

Table 8 *Output variable AC suffixes*

Suffix	Meaning of output variables
none	magnitude
DB	magnitude in decibels
G	group delay ($-d\text{PHASE}/d\text{FREQUENCY}$)
I	imaginary part
M	magnitude
P	phase in degrees
R	real part

Table 9 *Device names for two-terminal device types*

Two-terminal device type*	Device type letter
capacitor	C
diode	D
voltage-controlled voltage source**	E
current-controlled current source**	F
voltage-controlled current source**	G
current-controlled voltage source**	H
independent current source	I
inductor	L
resistor	R
voltage-controlled switch**	S
independent voltage source	V
current-controlled switch**	W

* The pin name for two-terminal devices is either 1 or 2.

** The controlling inputs for these devices are not considered terminals.

Table 10 *Terminal IDs by three & four-terminal device type*

Three & four-terminal device type	Device type letter	Terminal IDs
GaAs MOSFET	B	D (drain) G (gate) S (source)
Junction FET	J	D (drain) G (gate) S (source)
MOSFET	M	D (drain) G (gate) S (source) B (bulk, substrate)
Bipolar transistor	Q	C (collector) B (base) E (emitter) S (substrate)
transmission line	T	A (<i>near</i> side) B (<i>far</i> side)
IGBT	Z	C (collector) G (gate) E (emitter)

Table 11 *Noise types by device type*

Device type	Noise types*	Meaning
B (GaAsFET)	FID	flicker noise
	RD	thermal noise associated with RD
	RG	thermal noise associated with RG
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
D (diode)	FID	flicker noise
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
J (JFET)	FID	flicker noise
	RD	thermal noise associated with RD
	RG	thermal noise associated with RG
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
M (MOSFET)	FID	flicker noise
	RB	thermal noise associated with RB
	RD	thermal noise associated with RD
	RG	thermal noise associated with RG
	RS	thermal noise associated with RS
	SID	shot noise
	TOT	total noise
Q (BJT)	FIB	flicker noise
	RB	thermal noise associated with RB
	RC	thermal noise associated with RC
	RE	thermal noise associated with RE
	SIB	shot noise associated with base current
	SIC	shot noise associated with collector current
	TOT	total noise
R (resistor)	TOT	total noise
Iswitch	TOT	total noise
Vswitch	TOT	total noise

* These variables report the contribution of the specified device's noise to the total output noise in units of V^2/Hz . This means that the sum of all device noise contributions is equal to the total output noise in V^2/Hz , $\text{NTOT}(\text{ONoise})$.

Analog trace expressions

Trace expression aliases

Analog trace expressions vary from the output variables used in simulation analyses because analog net values can be specified by:

`<output variable>[:display name]`

as opposed to the `<output variable>` format used in analyses. With this format, the analog trace expression can be displayed in the analog legend with an optional alias.

Arithmetic functions

Arithmetic expressions of analog output variables use the same operators as those used in simulation analyses (by means of part property definitions in Capture). You can also include intrinsic functions in expressions. The intrinsic functions available for trace expressions are similar to those available for PSpice math expressions, but with some differences, as shown in [Table 12](#). A complete list of PSpice arithmetic functions can be found in [Table 10 on page 3-71](#).

Table 12 Analog arithmetic functions for trace expressions

Probe function	Description	Available in PSpice?
ABS(x)	x	YES
SGN(x)	+1 (if x>0), 0 (if x=0), -1 (if x<0)	YES
SQRT(x)	$x^{1/2}$	YES
EXP(x)	e^x	YES
LOG(x)	$\ln(x)$	YES
LOG10(x)	$\log(x)$	YES
M(x)	magnitude of x	YES
P(x)	phase of x (degrees)	YES
R(x)	real part of x	YES
IMG(x)	imaginary part of x	YES

Table 12 *Analog arithmetic functions for trace expressions*

Probe function	Description	Available in PSpice?
G(x)	group delay of x (seconds)	NO
PWR(x,y)	$ x ^y$	YES
SIN(x)	$\sin(x)$	YES
COS(x)	$\cos(x)$	YES
TAN(x)	$\tan(x)$	YES
ATAN(x)	$\tan^{-1}(x)$	YES
ARCTAN(x)		
d(x)	derivative of x with respect to the x-axis variable	YES*
s(x)	integral of x over the range of the x-axis variable	YES**
AVG(x)	running average of x over the range of the x-axis variable	NO
AVGX(x,d)	running average of x from X_axis_value(x)-d to X_axis_value(x)	NO
RMS(x)	running RMS average of x over the range of the x-axis variable	NO
DB(x)	magnitude in decibels of x	NO
MIN(x)	minimum of the real part of x	NO
MAX(x)	maximum of the real part of x	NO

* In PSpice, this function is called DDT(x).

** In PSpice, this function is called SDT(x).

Note *For AC analysis, PSpice uses complex arithmetic to evaluate trace expressions. If the result of the expression is complex, then its magnitude is displayed.*

Rules for numeric values suffixes

Explicit numeric values are entered in trace expressions in the same form as in simulation analyses (by means of part properties in Capture), with the following exceptions:

- Suffixes M and MEG are replaced with m (milli, 1E-3) and M (mega, 1E+6), respectively.

Example: V(5) and v(5) are equivalent in trace expressions.

Example: The quantities 2e-3, 2mV, and .002v all have the same numerical value. For axis labeling purposes, PSpice recognizes that the second and third forms are in volts, whereas the first is dimensionless.

PSpice also knows that $W=V \cdot A$, $V=W/A$, and $A=W/V$. So, if you add this trace:

V(5)*ID(M13)

the axis values are labeled with W.

For a demonstration of analog trace presentation, see [Analog example on page 13-341](#).

- MIL and mil are not supported.
- With the exception of the m and M scale suffixes, PSpice is not case sensitive; therefore, upper and lower case characters are equivalent.

Unit suffixes are *only* used to label the axis; they never affect the numerical results. Therefore, it is always safe to leave off a unit suffix.

The units to use for trace expressions are shown in [Table 13](#).

Table 13 Output units for trace expressions

Symbol	Unit
V	volt
A	amps
W	watt
d	degree (of phase)
s	second
Hz	hertz

Other output options

14

Chapter overview

This chapter describes how to output results in addition to those normally written to the data file or output file.

- [Viewing analog results in the PSpice window on page 14-368](#) explains how to monitor the numerical values for voltages or currents on up to three nets in your circuit as the simulation proceeds.
- [Writing additional results to the PSpice output file on page 14-369](#) explains how to generate additional line plots and tables of voltage and current values to the PSpice output file.

Viewing analog results in the PSpice window

Capture provides a special WATCH1 part that lets you monitor voltage values for up to three nets in your schematic as a DC sweep, AC sweep or transient analysis proceeds. Results are displayed in PSpice.

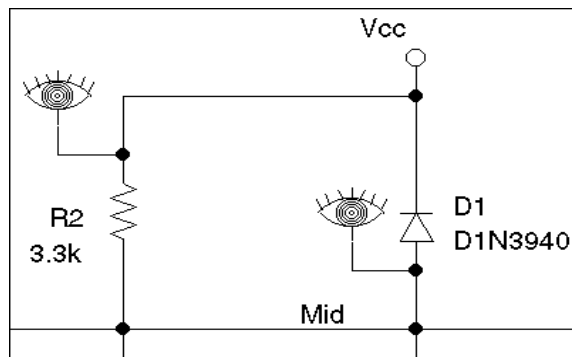
To display voltage values in the PSpice window



- 1 Place and connect a WATCH1 part (from the PSpice library SPECIAL.OLB) on an analog net.
- 2 Double-click the WATCH1 part instance to display the Parts spreadsheet.
- 3 In the ANALYSIS property column, type DC, AC, or TRAN (transient) for the type of analysis results you want to see.
- 4 Enter values in the LO and HI properties columns to define the lower and upper bounds, respectively, on the values you expect to see on this net.
- 5 Repeat steps **1** through **4** for up to two more WATCH1 instances.
- 6 Start the simulation.

If the results move outside of the specified bounds, PSpice pauses the simulation so that you can investigate the behavior.

For example, in the schematic fragment shown below, WATCH1 parts are connected to the Mid and Vcc nets. After starting the simulation, PSpice displays voltages on the Mid and Vcc nets.



Writing additional results to the PSpice output file

Capture provides special parts that let you save additional simulation results to the PSpice output file as either line-printer plots or tables.

To view the PSpice output file after having run a simulation:

- 1 From the Simulation menu, choose Examine Output.

Generating plots of voltage and current values

You can generate voltage and current line-printer plots for any DC sweep, AC sweep, or transient analysis.

To generate plots of voltage or current to the output file

- 1 Place and connect any of the following parts (from the PSpice library SPECIAL.OLB).

Table 14

Use this part...	To plot this...
VPLOT1	Voltage on the net that the part terminal is connected to.
VPLOT2	Voltage differential between the two nets that the part terminals are connected to.
IPLOT	Current through a net. (Insert this part in series, like a current meter.)



- 2 Double-click the part instance to display the Parts spreadsheet.
- 3 Click the property name for the analysis type that you want plotted: DC, AC, or TRAN.
- 4 In the columns for the analysis type that you want plotted (DC, AC or TRAN), type any non-blank value such as Y, YES or 1.

If you do not enable a format, PSpice defaults to MAG.

- 5 If you selected the AC analysis type, enable an output format:
 - a Click the property name for one of the following output formats: MAG (magnitude), PHASE, REAL, IMAG (imaginary), or DB.
 - b Type any non-blank value such as Y, YES or 1.
 - c Repeat the previous steps (a) and (b) for as many AC output formats as you want to see plotted.
- 6 Repeat steps 2 through 5 for any additional analysis types you want plotted.

Note If you do not enable an analysis type, PSpice reports the transient results.




Generating tables of voltage and current values

You can generate tables of voltage and current values on nets for any DC sweep, AC sweep, or transient analysis.

To generate tables of voltage or current to the output file

- 1 Place and connect any of the following parts (from the PSpice library SPECIAL.OLB).

Table 15

Use this part...	To tabulate this...
 VPRINT1	Voltage on the net that the part terminal is connected to.
 VPRINT2	Voltage differential between the two nets that the part terminals are connected to.
 IPRINT	Current through a cut in the net. (Insert this part in series, like a current meter.)

- 2 Double-click the part instance to display the Parts spreadsheet.

- 3 Click the property name for the analysis type that you want tabulated: DC, AC, or TRAN.
- 4 In the columns for the analysis type that you want plotted (DC, AC or TRAN), type any non-blank value such as Y, YES or 1.
- 5 If you selected the AC analysis type, enable an output format.
 - a Click the property name for one of the following output formats: MAG (magnitude), PHASE, REAL, IMAG (imaginary), or DB.
 - b Type any non-blank value such as Y, YES or 1.
 - c Repeat the previous steps (a) and (b) for as many AC output formats as you want to see tabulated.
- 6 Repeat steps 2 through 5 for any additional analysis types you want plotted.

If you do not enable a format, PSpice defaults to MAG.

Note *If you do not enable an analysis type, PSpice reports the transient results.*

Setting initial state

A

Appendix overview

This appendix includes the following sections:

- [Save and load bias point on page A-374](#)
- [Setpoints on page A-376](#)
- [Setting initial conditions on page A-378](#)

If the circuit uses high gain components, or if the circuit's behavior is nonlinear around the bias point, this feature is not useful.

Save and load bias point

Save Bias Point and Load Bias Point are used to save and restore bias point calculations in successive PSpice simulations. Saving and restoring bias point calculations can decrease simulation times when large circuits are run multiple times and can aid convergence.

Save/Load Bias Point affect the following types of analyses:

- transient
- DC
- AC

Save bias point

Save bias point is a simulation control function that allows you to save the bias point data from one simulation for use as initial conditions in subsequent simulations. Once bias point data is saved to a file, you can use the load bias point function to use the data for another simulation.

To use save bias point

See [Setting up analyses on page 7-197](#) for a description of the Analysis Setup dialog box.

- 1 In the Simulation Settings dialog box, click the Analysis tab.
- 2 Under Options, select Save Bias Point.
- 3 Complete the Save Bias Point dialog box.
- 4 Click OK.

Load bias point

Load bias point is a simulation control function that allows you to set the bias point as an initial condition. A common reason for giving PSpice initial conditions is to select one out of two or more stable operating points (set or reset for a flip-flop, for example).

To use load bias point

- 1 Run a simulation using the Save Bias Point option in the Simulation Settings dialog box.
- 2 Before running another simulation, click the Analysis tab in the Simulation Settings dialog box.
- 3 Under Options, select Load Bias Point.
- 4 Specify a bias point file to load. Include the path if the file is not located in your working directory, or use the Browse button to find the file.
- 5 Click OK.

See [Setting up analyses on page 7-197](#) for a description of the Analysis Setup dialog box.

Setpoints

Pseudocomponents that specify initial conditions are called setpoints. These apply to the analog portion of your circuit.

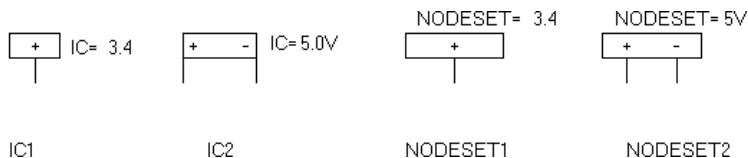


Figure A-1 *Setpoints.*

The example in Figure A-1 includes the following:

- IC1 a one-pin symbol that allows you to set the initial condition on a net for both small-signal and transient bias points
- IC2 a two-pin symbol that allows you to set initial condition between two nets

Using IC symbols sets the initial conditions for the bias point only. It does not affect the DC sweep. If your circuit design contains both an IC symbol and a NODESET symbol for the same net, the NODESET symbol is ignored.

To specify the initial condition, edit the value of the VALUE property to the desired initial condition. PSpice attaches a voltage source with a 0.0002 ohm series resistance to each net to which an IC symbol is connected. The voltages are clamped this way for the entire bias point calculation.

NODESET1 is a one-pin symbol which helps calculate the bias point by providing a initial guess for some net. NODESET2 is a two-pin symbol which helps calculate the bias point between two nets. Some or all of the circuit's nets may be given an initial guess. NODESET symbols are effective for the bias point (both small-signal and transient bias points) and for the first step of the DC sweep. It has no effect during the rest of the DC sweep or during the transient analysis itself.

Unlike the IC pseudocomponents, NODESET provides only an initial guess for some net voltages. It does not clamp those nodes to the specified voltages. However, by providing an initial guess, NODESET symbols may be used to break the tie (in a flip-flop, for instance) and make it come up in a desired state. To guess at the bias point, enter the initial guess in the Value text box for the VALUE property. PSpice attaches a voltage source with a 0.0002 ohm series resistance to each net to which an IC symbol is connected.

These pseudocomponents are netlisted as PSpice .IC and .NODESET commands. Refer to these commands in the online *OrCAD PSpice A/D Reference Manual* for more information. Setpoints can be created for inductor currents and capacitor voltages using the IC property described in [Setting initial conditions on page A-378](#).

Setting initial conditions

The IC property allows initial conditions to be set on capacitors and inductors. These conditions are applied during all bias point calculations. However, if you select the Skip Initial Transient Solution check box in the Transient Analysis Setup dialog box, the bias point calculation is skipped and the simulation proceeds directly with transient analysis at TIME=0. Devices with the IC property defined start with the specified voltage or current value; however, all other such devices have an initial voltage or current of 0.

Note *Skipping the bias point calculation can make the transient analysis subject to convergence problems.*

See [Setpoints on page A-376](#) for more information about IC1 and IC2.

Applying an IC property for a capacitor has the same effect as applying one of the pseudocomponents IC1 or IC2 across its nodes. PSpice attaches a voltage source with a 0.002 ohm series resistance in parallel with the capacitor. The IC property allows the user to associate the initial condition with a device, while the IC1 and IC2 pseudocomponents allow the association to be with a node or node pair.

In the case of initial currents through inductors, the association is only with a device, and so there are no corresponding pseudocomponents. The internal implementation is analogous to the capacitor. PSpice attaches a current source with a 1 Gohm parallel resistance in series with the inductor.

Convergence and “time step too small errors”

B

Appendix overview

This appendix discusses common errors and convergence problems in PSpice.

- [Introduction on page B-380](#)
- [Bias point and DC sweep on page B-385](#)
- [Transient analysis on page B-388](#)
- [Diagnostics on page B-393](#)

Introduction

In order to calculate the bias point, DC sweep and transient analysis for analog devices PSpice must solve a set of nonlinear equations which describe the circuit's behavior. This is accomplished by using an iterative technique—the Newton-Raphson algorithm—which starts by having an initial approximation to the solution and iteratively improves it until successive voltages and currents converge to the same result.

In a few cases PSpice cannot find a solution to the nonlinear circuit equations. This is generally called a “convergence problem” because the symptom is that the Newton-Raphson repeating series cannot converge onto a consistent set of voltages and currents. The following discussion gives some background on the algorithms in PSpice and some guidelines for avoiding convergence problems.

The AC and noise analyses are linear and do not use an iterative algorithm, so the following discussion does not apply to them.

The transient analysis has the additional possibility of being unable to continue because the time step required becomes too small from something in the circuit moving too fast. This is also discussed below.

Newton-Raphson requirements

The Newton-Raphson algorithm is *guaranteed to converge to a solution*. However, this guarantee has some conditions:

- 1 The nonlinear equations must have a solution.
- 2 The equations must be continuous.
- 3 The algorithm needs the equations' derivatives.
- 4 The initial approximation must be close enough to the solution.

Each of these can be taken in order. Remember that the PSpice algorithms are used in computer hardware that

has finite precision and finite dynamic range that produce these limits:

- Voltages and currents in PSpice are limited to +/-1e10 volts and amps.
- Derivatives in PSpice are limited to 1e14.
- The arithmetic used in PSpice is double precision and has 15 digits of accuracy.

Is there a solution?

Yes, for any physically realistic circuit. However, it is not difficult to set up a circuit that has no solution within the limits of PSpice numerics. Consider, for example, a voltage source of one megavolt connected to a resistor of one micro-ohm. This circuit does not have a solution within the dynamic range of currents (+/- 1e10 amps).

Here is another example:

```
V1      1,      0      5v
D1      1,      0      DMOD
.MODEL          DMOD(IS=1e-16)
```

The problem here is that the diode model has no series resistance. It can be shown that the current through a diode is:

$$I = IS * e^{V / (N * k * T)}$$

N defaults to one and $k * T$ at room temperature is about .025 volts. So, in this example the current through the diode would be:

$$I = 1e-16 * e^{200} = 7.22e70 \text{ amps}$$

This circuit also does not have a solution within the limits of the dynamic range of PSpice. In general, be careful of components without limits built into them. Extra care is needed when using the expressions for controlled sources (such as for behavioral modeling). It is easy to write expressions with very large values.

To find out more about the diode equations, refer to the *Analog Devices* chapter in the online *OrCAD PSpice A/D Reference Manual*.

Are the Equations Continuous?

The device equations built into PSpice are continuous. The functions available for behavioral modeling are also continuous (there are several functions, such as `int(x)`, which cannot be added because of this). So, for physically realistic circuits the equations can also be continuous. Exceptions that come are usually from exceeding the limits of the numerics in PSpice. This example tries to approximate an ideal switch using the diode model:

```
.MODEL DMOD(IS=1e-16 N=1e-6)
```

The current through this diode is:

$$I = 1e-16 * e^{V / (N * .025)} = 1e-16 * e^{V / 25e-9}$$

Avoid unrealistic model parameters. Behavioral modeling expressions need extra care.

Because the denominator in the exponential is so small, the current I is essentially zero for $V < 0$ and almost infinite for $V > 0$. Even if there are external components that limit the current, the “knee” of the diode's I-V curve is so sharp that it is almost a discontinuity.

Are the derivatives correct?

The device equations built into PSpice include the derivatives, and these are correct. Depending on the device, the physical meaning of the derivatives is small-signal conductance, transconductance or gain.

Unrealistic model parameters can exceed the limit of $1e14$, but it requires some effort. The main thing to look at is the behavioral modeling expressions, especially those having denominators.

Is the initial approximation close enough?

Newton-Raphson is guaranteed to converge only if the analysis is started close to the answer. Also, there is no measurement that can tell how close is close enough.

PSpice gets around this by making heavy use of continuity. Each analysis starts from a known solution and uses a variable step size to find the next solution. If the next solution does not converge PSpice reduces the step size, falls back and tries again.

Bias point

The hardest part of the whole process is getting started, that is, finding the bias point. PSpice first tries with the power supplies set to 100%. A solution is not guaranteed, but most of the time the PSpice algorithm finds one. If not, then the power supplies are cut back to almost zero. They are cut to a level small enough that *all nonlinearities are turned off*. When the circuit is linear a solution can be found (very near zero, of course). Then, PSpice works its way back up to 100% power supplies using a variable step size.

Once a bias point is found the transient analysis can be run. It starts from a known solution (the bias point) and steps forward in time. The step size is variable and is reduced as needed to find further solutions.

DC sweep

The DC sweep uses a hybrid approach. It uses the bias point algorithm (varying the power supplies) to get started. For subsequent steps it uses the previous solution as the initial approximation. The sweep step is not variable, however. If a solution cannot be found at a step then the bias point algorithm is used for that step.

The whole process relies heavily on continuity. It also requires that the circuit be linear when the supplies are turned off.

STEPGMIN

An alternative algorithm is GMIN stepping. This is not obtained by default, and is enabled by specifying the circuit analysis option STEPGMIN (either using `.OPTION STEPGMIN` in the netlist, or by making the appropriate choice from the Analysis/Setup/Options menu). When enabled, the GMIN stepping algorithm is applied after the circuit fails to converge with the power supplies at 100 percent, and if GMIN stepping also fails, the supplies are then cut back to almost zero.

GMIN stepping attempts to find a solution by starting the repeating cycle with a large value of GMIN, initially $1.0e10$ times the nominal value. If a solution is found at this setting it then reduces GMIN by a factor of 10, and tries again. This continues until either GMIN is back to the nominal value, or a repeating cycle fails to converge. In the latter case, GMIN is restored to the nominal value and the power supplies are stepped.

Bias point and DC sweep

Power supply stepping

As previously discussed, PSpice uses a proprietary algorithm which finds a continuous path from zero power supplies levels to 100%. It starts at almost zero (.001%) power supplies levels and works its way back up to the 100% levels. The minimum step size is $1e-6$ (.0001%). The first repeating series of the first step *starts at zero for all voltages*.

Semiconductors

Model parameters

The first consideration for semiconductors is to avoid physically unrealistic model parameters. Remember that as PSpice steps the power supplies up it has to step carefully through the turn on transition for each device. In the diode example above, for the setting $N=1e-6$, the knee of the I-V curve would be too sharp for PSpice to maintain its continuity within the power supply step size limit of $1e-6$.

Unguarded p-n junctions

A second consideration is to avoid “unguarded” p-n junctions (no series resistance). The above diode example also applies to the p-n junctions inside bipolar transistors, MOSFETs (drain-bulk and source-bulk), JFETs and GaAsFETs.

No leakage resistance

A third consideration is to avoid situations which could have an ideal current source pushing current into a reverse-biased p-n junction without a shunt resistance. Since p-n junctions in PSpice have (almost) no leakage resistance and would cause the junction's voltage to go beyond 1e10 volts.

The model libraries which are part of PSpice follow these guidelines.

Typos can cause unrealistic device parameters. The following MOSFET:

```
M1 3, 2, 1, 0 MMOD L=5 W=3
```

has a length of five meters and a width of three meters instead of micrometers. It should have been:

```
M1 3, 2, 1, 0 MMOD L=5u W=3u
```

PSpice flags an error for L too large, but cannot for W because power MOSFETs are so interdigitated (a zipper-like trace) that their effective W can be very high. The LIST option can show this kind of problem. When the devices are listed in the output file their values are shown in scientific notation making it easy to spot unusual values.

Switches

PSpice switches have gain in their transition region. If several are cascaded then the cumulative gain can easily exceed the derivative limit of 1e14. This can happen when modeling simple logic gates using totem-pole switches and there are several gates in cascaded in series. Usually a cascade of two switches works but three or more can cause trouble.

Behavioral modeling expressions

Range limits

Voltages and currents in PSpice are limited to the range $\pm 1e10$. Care must be taken that the output of expressions fall within this range. This is especially important when one is building an electrical analog of a mechanical, hydraulic or other type of system.

Source limits

Another consideration is that the controlled sources must turn off when the supplies are almost 0 (.001%). There is special code in PSpice which “squelsches” the controlled sources in a continuous way near 0 supplies. However, care should still be taken using expressions that have denominators. Take, for example, a constant power load:

```
GLOAD 3, 5 VALUE = {2Watts/V(3,5)}
```

The first repeating series starts with $V(3,5) = 0$ and the current through GLOAD would be infinite (actually, the code in PSpice which does the division clips the result to a finite value). The “squelching” code is required to be a smooth and well-behaved function.

Note *The “squelching” code cannot be “strong” enough to suppress dividing by 0.*

The result is that GLOAD does not turn off near 0 power supplies. A better way is described in the application note Modeling Constant Power Loads. The “squelching” code is sufficient for turning off all expressions except those having denominators. In general, though, it is good practice to constrain expressions having the LIMIT function to keep results within physically realistic bounds.

Example: A first approximation to an opamp that has an open loop gain of 100,000 is:

```
VOPAMP 3, 5 VALUE = {V(in+,in-)*1e5}
```

This has the undesirable property that there is no limit on the output. A better expression is:

```
VOPAMP 3, 5 VALUE =  
+ {LIMIT(V(in+,in-)*1e5,15v,-15v)}
```

where the output is limited to +/- 15 volts.

Transient analysis

The transient analysis starts using a known solution - the bias point. It then uses the most recent solution as the first guess for each new time point. If necessary, the time step is cut back to keep the new time point close enough that the first guess allows the Newton-Raphson repeating series to converge. The time step is also adjusted to keep the integration of charges and fluxes accurate enough.

In theory the same considerations which were noted for the bias point calculation apply to the transient analysis. However, in practice they show up during the bias point calculation first and, hence, are corrected before a transient analysis is run.

The transient analysis can fail to complete if the time step gets too small. This can have two different effects:

- 1 The Newton-Raphson iterations would not converge even for the smallest time step size, or
- 2 Something in the circuit is moving faster than can be accommodated by the minimum step size.

The message PSpice puts into the output file specifies which condition occurred.

Skipping the bias point

The SKIPBP option for the transient analysis skips the bias point calculation. In this case the transient analysis has no known solution to start from and, therefore, is not assured of converging at the first time point. Because of this, its use is not recommended. Its inclusion in PSpice is to maintain compatibility with UC Berkeley SPICE. SKIPBP has the same meaning as UIC in Berkeley SPICE. UIC is not needed in order to specify initial conditions.

The dynamic range of TIME

TIME, the simulation time during transient analysis, is a double precision variable which gives it about 15 digits of accuracy. The dynamic range is set to be 15 digits minus the number of digits of accuracy required by RELTOL. For a default value of $RELTOL = .001$ (.1% or 3 digits) this gives $15 - 3 = 12$ digits. This means that the minimum time step is the overall run time (TSTOP) divided by $1e12$. The dynamic range is large but finite.

It is possible to exceed this dynamic range in some circuits. Consider, for example, a timer circuit which charges up a 100uF capacitor to provide a delay of 100 seconds. At a certain threshold a comparator turns on a power MOSFET. The overall simulation time is 100 seconds. For default RELTOL this gives us a minimum time step of 100 picoseconds. If the comparator and other circuitry has portions that switch in a nanosecond then PSpice needs steps of less than 100 picoseconds to calculate the transition accurately.

Failure at the first time step

If the transient analysis fails at the first time point then usually there is an unreasonably large capacitor or inductor. Usually this is due to a typographical error. Consider the following capacitor:

```
C 1 3, 0 10uf
```

“10” (has the letter O) should have been “10.” This capacitor has a value of one farad, not 10 microfarads. An easy way to catch these is to use the LIST option (on the .OPTIONS command).

LIST

The LIST option can echo back all the devices into the output file *that have their values in scientific notation*.

That makes it easy to spot any unusual values. This kind of problem does not show up during the bias point calculation because capacitors and inductors do not participate in the bias point.

Similar comments apply to the parasitic capacitance parameters in transistor (and diode) models. These are normally echoed to the output file (the NOMOD option suppresses the echo but the default is to echo). As in the LIST output, the model parameters are echoed in scientific notation making it easy to spot unusual values. A further diagnostic is to ask for the detailed operating bias point (.TRAN/OP) information.

.TRAN/OP

This lists the small-signal parameters for each semiconductor device including the calculated parasitic capacitances.

Parasitic capacitances

It is important that switching times be nonzero. This is assured if devices have parasitic capacitances. The semiconductor model libraries in PSpice have such capacitances. If switches and/or controlled sources are used, then care should be taken to assure that no sections of circuitry can try to switch in zero time. In practice this means that if any positive feedback loops exist (such as a Schmidt trigger built out of switches) then such loops should include capacitances.

Another way of saying all this is that during transient analysis the circuit equations must be continuous over time (just as during the bias point calculation the equations must be continuous with the power supply level).

Inductors and transformers

While the impedance of capacitors gets lower at high frequencies (and small time steps) the impedance of inductors gets higher.

Note The inductors in PSpice have an infinite bandwidth.

Real inductors have a finite bandwidth due to eddy current losses and/or skin effect. At high frequencies the effective inductance drops. Another way to say this is that physical inductors have a frequency at which their Q begins to roll off. The inductors in PSpice have no such limit. This can lead to very fast spikes as transistors (and diodes) connected to inductors turn on and off. The fast spikes, in turn, can force PSpice to take unrealistically small time steps.

Note OrCAD recommends that all inductors have a parallel resistor (series resistance is good for modeling DC effects but does not limit the inductor's bandwidth).

The parallel resistor gives a good model for eddy current loss and limits the bandwidth of the inductor. The size of

resistor should be set to be equal to the inductor's impedance at the frequency at which its Q begins to roll off.

Example: A common one millihenry iron core inductor begins to roll off at no less than 100KHz. A good resistor value to use in parallel is then $R = 2 * \pi * 100e3 * .001 = 628$ ohms. Below the roll-off frequency the inductor dominates; above it the resistor does. This keeps the width of spikes from becoming unreasonably narrow.

Bipolar transistors substrate junction

The UC Berkeley SPICE contains an unfortunate convention for the substrate node of bipolar transistors. The collector-substrate p-n junction has *no DC component*. If the capacitance model parameters are specified (e.g., CJS) then the junction has (voltage-dependent) capacitance but no DC current. This can lead to a sneaky problem: if the junction is inadvertently forward-biased it can create a very large capacitance. The capacitance goes as a power of the junction voltage. Normal junctions cannot sustain much forward voltage because a large current flows. The collector-substrate junction is an exception because it has no DC current.

If this happens it usually shows up at the first time step. It can be spotted turning on the detailed operating point information (.TRAN/OP) and looking at the calculated value of CJS for bipolar transistors. The whole problem can be prevented by using the PSpice model parameter ISS. This parameter “turns on” DC current for the substrate junction.

Diagnostics

If PSpice encounters a convergence problem it inserts into the output file a message that looks like the following.

```
ERROR -- Convergence problem in transient analysis at Time = 7.920E-03
```

```
Time step = 47.69E-15, minimum allowable step size = 300.0E-15
```

```
These voltages failed to converge:
```

```
V(x2.23) = 1230.23 / -68.4137
V(x2.25) = -1211.94 / 86.6888
```

```
These supply currents failed to converge:
```

```
I(X2.L1) = -36.6259 / 2.25682
I(X2.L2) = -36.5838 / 2.29898
```

```
These devices failed to converge:
```

```
X2.DCR3 X2.DCR4 x2.ktr X2.Q1 X2.Q2
```

```
Last node voltages tried were:
```

NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE	NODE	VOLTAGE
			E				
(1)	25.2000	(3)	4.0000	(4)	0.0000	(6)	25.2030
(x2.23)	1230.200	(X2.24)	9.1441	(x2.25)	-1211.900	(X2.26)	256.970
)	0))	0)	0
(X2.28)	-206.610	(X2.29)	75.487	(X2.30)	-25.0780	(X2.31)	26.2810
)	0)	0))	
(X3.34)	1.771E-0	(X3.35)	1.0881	(X3.36)	.4279	(X2.XU1.6)	1.2636
)	6)))	

The message always includes the banner (ERROR -- convergence problem ...) and the trailer (Last node voltages tried were ...). It cannot include all three of the middle blocks.

The `Last node voltages tried...` trailer shows the voltages tried at the last Newton-Raphson iteration. If any of the nodes have unreasonable large values this is a clue that these nodes are related to the problem. “These voltages failed to converge” lists the specific nodes which did not settle onto consistent values. It also shows their values for the last two iterations. “These supply currents failed converge” does the same for currents through voltage sources and inductors. If any of the listed numbers are $\pm 1e10$ then that is an indication that the value is being clipped from an unreasonable value. Finally, “These devices failed to converge” shows devices whose terminal currents or core fluxes did not settle onto consistent values.

The message gives a clue as to the part of the circuit which is causing the problem. Looking at those devices and/or nodes for the problems discussed above is recommended.

Index

A

ABM

- ABM part templates, 152
- ABM.OLB, 149
- basic controlled sources, 192
- cautions and recommendations for simulation, 186
- control system parts, 153
- custom parts, 192
- frequency domain device models, 181
- frequency domain parts, 181, 187
- instantaneous models, 176, 186
- overview, 148
- placing and specifying ABM parts, 150
- PSpice A/D-equivalent parts, 174–175
- signal names, 147
- simulation accuracy, 191
- syntax, 175
- triode modeling example, 171

AC stimulus property, 234

AC sweep analysis, 196, 231–232

- about, 232
- displaying simulation results, 39
- example, 37, 237
- introduction, 4

noise analysis, 196, 241

setup, 37, 232, 235

stimulus, 233

treatment of nonlinear devices, 239

ACMAG stimulus property, 234

ACPHASE stimulus property, 234

adding a stimulus, 33

AGND ground part, 83

analog behavioral modeling, *see* ABM

analog parts

- basic components (ABM), 153, 155

- basic controlled sources (ABM), 192

- behavioral, 66

- bipolar transistors, 95, 204, 362–363

- breakout, 65

- capacitors, 202

- Chebyshev filters, 153, 157, 190, 301

- Darlington model transistors, 95

- diodes, 95, 202, 363

- expression parts (ABM), 154, 168

- frequency table parts (ABM), 174, 183, 190

- GaAsFET, 203, 362–363

- IGBT, 95, 204, 362

- inductors, 202

- integrators and differentiators (ABM), 153, 160

- JFET, 95, 203, 362–363

- Laplace transform (ABM), 154, 164, 174, 181, 187
- limiters (ABM), 153, 156
- math functions (ABM), 154, 167
- mathematical expressions (ABM), 174
- MOSFET, 95, 204, 362–363
- nonlinear magnetic core, 95
- opamp, 95
- passive, 64
- PSpice A/D-equivalent parts (ABM), 174
- resistors, 203, 363
- switch, 363
- table look-up (ABM), 153, 160, 174, 179
- transmission lines, 204, 362
- vendor-supplied, 61
- voltage comparator, 95
- voltage reference, 95
- voltage regulator, 95

analyses

- AC sweep, 37, 196, 231–232
- bias point detail, 22, 196
- DC sensitivity, 196, 228
- DC sweep, 26, 196, 214
- execution order, 198
- Fourier, 196
- frequency response, 196
- Monte Carlo, 196, 289
- noise, 196, 241
- overview, 3
- parametric, 42, 196, 272
- performance analysis, 49
- sensitivity/worst-case, 196, 306
- setup, 197
- small-signal DC transfer, 196, 225
- temperature, 196, 281
- transient, 32, 196
- types, 196

- approximation, problems, 383

B

- basic components (ABM), 153, 155
- basic controlled sources (ABM), 192
- behavioral modeling expressions, 387
- behavioral parts, 66
- bias point
 - convergence analysis, 389
 - save/restore, 374
 - skipping, 389
- bias point detail analysis, 196

- example, 22
 - introduction, 3
- bipolar transistors, 95, 204, 362–363
 - problems, 392
- Bode plot, 4, 40

C

- capacitors, 202
- Chebyshev filters, 153, 157, 190, 301
- circuit file (.CIR), 10
 - simulating multiple circuits, 207
- color settings for waveform analysis, 324
- comparator, 95
- continuous equations
 - problems, 382
- control system parts (ABM), 153
- controlled sources, 174, 192
- convergence analysis
 - bias point, 389
- convergence problems, 379
 - approximations, 383
 - behavioral modeling expressions, 387
 - bias point, 385
 - bipolar transistors, 392
 - continuous equations, 382
 - DC sweep, 385
 - derivatives, 382
 - diagnostics, 393
 - dynamic range of time, 389
 - inductors and transformers, 391
 - Newton-Raphson requirements, 380
 - parasitic capacitances, 391
 - semiconductors, 385
 - switches, 386
 - transient analysis, 388
- Create Subcircuit command, 91, 115
- current source, controlled, 174, 192
- cursors, waveform analysis, 350
- custom part creation for models, 133
 - using the Model Editor, 100, 131

D

- Darlington model transistors, 95
- DC analyses
 - displaying simulation results, 28

see also DC sweep analysis, bias point detail analysis, small-signal DC transfer analysis, DC sensitivity analysis

DC sensitivity analysis, 196, 228
introduction, 3

DC stimulus property, 219

DC sweep analysis, 196, 214
about, 216
curve families, 221
example, 26
introduction, 3
nested, 219
setting up, 26
stimulus, 218

derivative
problems, 382

design
preparing for simulation, 9, 56

device noise, 242, 245
total, 245

diagnostic problems, 393

differentiators (ABM), 153, 160

digital parts
vendor-supplied, 61

digital simulation
waveform display, 344, 364

diodes, 95, 202, 363

dynamic range of time, 389

E

EGND ground part, 83

examples and tutorials
AC sweep analysis, 37, 237
analog waveform analysis, 339
bias point detail analysis, 22
DC sweep analysis, 26
example circuit creation, 16
frequency response vs. arbitrary parameter, 278
mixed analog/digital waveform analysis, 344
modeling a triode (ABM), 171
Monte Carlo analysis, 293
parametric analysis, 42
performance analysis, 49, 274
transient analysis, 32
using the Model Editor, 104, 114
worst-case analysis, 309

expression parts (ABM), 154, 168

expressions, 69–70

see also parameters

ABM, 174

functions, 71

specifying, 69

system variables, 73

waveform analysis, 364

F

files
generated by Capture, 10
user-configurable, 11
with simulation results, 14

flicker noise, 245

Fourier analysis, 196
introduction, 5

FREQUENCY output variable, 359

frequency response vs. arbitrary parameter, 278

frequency table parts (ABM), 174, 183, 190

functions
PSPice A/D, 71
waveform analysis, 364

G

GaAsFET, 203, 362–363

goal functions, 275
in performance analysis, 276
single data point, 276

grid spacing
part graphics, 136
part pins, 136

ground
missing, 83
missing DC path to, 84
parts, 60

group delay, 361

H

histograms, 301

hysteresis curves, 266

I

IAC stimulus part, 233

IC (property), 378

ICn initial conditions parts, 376

- IDC stimulus part, 74, 218
- IGBT, 95, 204, 362
- imaginary part, 361
- include files, 11
 - configuring, 13, 120
 - with model definitions, 121
- inductors, 202
 - problems, 391
- initial conditions, 374, 378
- input noise, total, 245
- instance models
 - and the Model Editor, 101, 112
 - changing model references, 117
 - editing, 103
 - reusing, 118
 - saving for global use instead using the Model Editor, 113
- integrators (ABM), 153, 160
- IPlot (write current plot part), 369
- IPrint (write current table part), 370
- ISRC stimulus part, 74, 218, 233
- ISTIM stimulus part, 76

J

- JFET, 95, 203, 362–363

L

- Laplace transform parts (ABM), 154, 164, 174, 181, 187–188
- libraries
 - configuring, 120
 - footprint, 13
 - model, 88
 - package, 13
 - part (.OLB), 13
 - searching for models, 121
 - see also* model libraries
- Library List, using the, 63
- limiters (ABM), 153, 156

M

- magnetic core, nonlinear, 95
- magnitude, 361
- markers, 334
 - displaying traces, 28

- for limiting waveform data file size, 334
 - for waveform display, 331
- math function parts (ABM), 154, 167
- mathematical expressions (ABM), 174
- mixed analog/digital circuits
 - waveform display, 344, 364
- Model Editor
 - about, 10, 110
 - analyzing model parameter effects, 97
 - changing
 - .MODEL definitions, 110
 - .SUBCKT definitions, 111
 - model names, 111
 - creating parts for models, 100, 131
 - custom, 133
 - example, 114
 - fitting models, 97
 - starting stand-alone, 99
 - starting from the schematic page editor, 101
 - supported device types, 95
 - testing and verifying models, 96
 - tutorial, 104
 - using data sheet information, 96
 - viewing performance curves, 98
 - ways to use, 94
- model editor
 - running from the schematic page editor, 111
- model libraries, 11, 88
 - adding to the configuration, 122
 - analog list of, 80
 - and duplicate model names, 122
 - configuration, 89
 - configured as include files, 121
 - configuring, 13, 81, 120
 - directory search path, 125
 - global vs. design, 89, 123
 - how PSpice searches them, 121
 - nested, 90
 - OrCAD-provided, 90
 - preparing for part creation, 130
 - search order, 121, 124
- MODEL property, 87, 138
- models
 - built-in, 2
 - changing associations to parts, 117
 - creating parts for
 - custom, 133
 - using the Model Editor, 100, 131
 - creating with the Model Editor, 110

- defined as
 - parameter sets, 87
 - subcircuits, 87, 115
- global vs. design, 89
- instance, 101, 112, 117–118
- organization, 88
- preparing for part creation, 130
- saving as design
 - using the Model Editor, 101
- saving as local
 - using the Model Editor, 111
- testing/verifying (Model Editor-created), 96
- tools to create, 91
- ways to create/edit, 92
- Monte Carlo analysis, 196, 289
 - collating functions, 287
 - histograms, 301
 - introduction, 7
 - model parameter values reports, 285
 - output control, 285
 - tutorial, 293
 - using the Model Editor, 114
 - waveform reports, 286
 - with temperature analysis, 288
- MOSFET, 95, 204, 362–363
- multiple y-axes, waveform analysis, 276

N

- netlist
 - failure to netlist, 58
 - file (.NET), 10
- Newton-Raphson requirements, 380
- NODESETn initial conditions parts, 376
- noise analysis, 196, 241
 - about, 4, 242
 - device noise, 242
 - flicker noise, 245
 - noise equations, 245
 - setup, 241, 243
 - shot noise, 245
 - thermal noise, 245
 - total output and input noise, 242
 - units of measure, 246
 - viewing results, 246
 - viewing simulation results, 245, 363
 - waveform analysis output variables, 245, 363
- noise units, 246
- non-causality, 188

- nonlinear
 - magnetic core, 95
- nonlinear devices
 - in AC sweep analysis, 239

O

- opamp, 95
- operators in expressions, 70
- options
 - RELTOL, 191
- output control parts, 60, 369
- output file (.OUT), 24
 - control parts, 369
 - tables and plots, 369
- output noise, total, 245
- output variables
 - arithmetic expressions, 364
 - noise (waveform analysis), 245, 363
 - PSpice A/D, 199
 - waveform analysis, 356
 - waveform analysis functions, 364

P

- PARAM global parameter part, 67
- parameters, 67
- parametric analysis, 196, 272
 - analyzing waveform families, 45
 - example, 42
 - frequency response vs. arbitrary parameter, 278
 - introduction, 6
 - performance analysis, 274
 - setting up, 43
 - temperature analysis, 196, 281
- parasitic capacitance, 391
- part wizard
 - using custom parts, 133
- parts
 - creating for models
 - using the Model Editor, 100, 131
 - creating new stimulus parts, 259
 - editing graphics, 135
 - grid spacing
 - graphics, 136
 - pins, 136
 - ground, 60
 - non-simulation, 140
 - output control, 60

- pins, 82, 136
- preparing model libraries for part creation, 130
- properties for simulation, 139
- saving as global
 - using the Model Editor, 100, 131
- simulation control, 60
- simulation properties, 129
- stimulus, 60
- ways to create for models, 129
- AGND (ground), 83
- BBREAK (GaAsFET), 65
- behavioral, 66
- breakout, 65
- C (capacitor), 64
- CBREAK (capacitor), 65
- creating for models
 - custom parts, 133
 - using the Model Editor, 131
- CVAR (capacitor), 64
- D (diode), 64
- DBREAK (diode), 65
- EGND (ground), 83
- finding, 62
- IAC (AC stimulus), 233
- ICn (initial conditions), 376
- IDC (DC stimulus), 74, 218
- ISRC (analog stimulus), 74, 218, 233
- ISTIM (transient stimulus), 76
- JBREAK (JFET), 65
- K_LINEAR (transformer), 64
- KBREAK (inductor coupling), 65
- KCOUPLEn (coupled transmission line), 64
- LBREAK (inductor), 65
- MBREAK (MOSFET), 65
- MODEL property, 138
- NODESETn, 376
- PARAM (global parameter), 67
- passive, 64
- QBREAK (bipolar transistor), 65
- R (resistor), 64
- RBREAK (resistor), 65
- RVAR (resistor), 64
- SBREAK (voltage-controlled switch), 65
- T (ideal transmission line), 64
- TBREAK (transmission line), 65
- TEMPLATE property, 140
- TnCOUPLEDx (coupled transmission line), 64
- unmodeled, 79
- VAC (AC stimulus), 75, 233
- VDC (DC stimulus), 74–75
- vendor-supplied, 61
- VEXP (transient stimulus), 75
- VPULSE (transient stimulus), 75
- VPWL (transient stimulus), 75
- VPWL_F_N_TIMES (transient stimulus), 76
- VPWL_F_RE_FOREVER (transient stimulus), 75
- VPWL_N_TIMES (transient stimulus), 76
- VPWL_RE_FOREVER (transient stimulus), 75
- VSFFM (transient stimulus), 76
- VSIN (transient stimulus), 76
- VSRC (analog stimulus), 74–75, 233
- VSTIM (analog stimulus), 75
- VSTIM (transient stimulus), 76
- WBREAK (current-controlled switch), 65
- XFRM_LINEAR (transformer), 64
- XFRM_NONLINEAR (transformer), 65
- ZBREAK (IGBT), 65
- ABMn and ABMn/I (ABM), 154, 168
- ABS (ABM), 154, 167
- ARCTAN (ABM), 154, 167
- ATAN (ABM), 154, 167
- BANDPASS (ABM), 153, 158
- BANDREJ (ABM), 153, 159
- CONST (ABM), 153, 155
- COS (ABM), 154, 167
- DIFF (ABM), 153, 155
- DIFFER (ABM), 153, 160
- E (ABM controlled analog source), 192
- EFREQ (ABM), 174, 183
- ELAPLACE (ABM), 174, 181
- EMULT (ABM), 174, 178
- ESUM (ABM), 174, 178
- ETABLE (ABM), 174, 179
- EVALUE (ABM), 174, 176–177
- EXP (ABM), 154, 167
- F (ABM controlled analog source), 192
- FTABLE (ABM), 153, 161
- G (ABM controlled analog source), 192
- GAIN (ABM), 153, 155
- GFREQ (ABM), 174, 183
- GLAPLACE (ABM), 174, 181
- GLIMIT (ABM), 153, 156
- GMULT (ABM), 174, 178
- GSUM (ABM), 174, 178
- GTABLE (ABM), 174, 179
- GVALUE (ABM), 174, 176–177
- H (ABM controlled analog source), 192
- HIPASS (ABM), 153, 158
- ICn (initial condition), 376
- INTEG (ABM), 153, 160

- LAPLACE (ABM), 154, 164
- LIMIT (ABM), 153, 156
- LOG (ABM), 154, 167
- LOG10 (ABM), 154, 167
- LOPASS (ABM), 153, 157
- MULT (ABM), 153, 155
- NODESETn (initial bias point), 376
- PWR (ABM), 154, 167
- PWRS (ABM), 154, 167
- SIN (ABM), 154, 167
- SOFTLIM (ABM), 153, 156
- SQRT (ABM), 154, 167
- SUM (ABM), 153, 155
- TABLE (ABM), 153, 160
- TAN (ABM), 154, 167
- performance analysis, 274
 - example, 49
 - goal functions, 275
- phase, 361
- plots in waveform analysis, 321
- power supplies
 - analog, 74
- Probe windows
 - plot update methods, 346
 - plots, 321–322
 - printing Probe windows, 323
 - scrolling, 346
 - setting colors, 324
 - trace data tables, 349
 - traces, displaying, 28
 - zoom regions, 344
- properties (part) for simulation, 139
- PSpice
 - default shortcut keys, 344
 - waveform analysis, 320
 - multiple y-axes, 276
- PSpice A/D
 - about, 2
 - expressions, 69
 - functions, 71
 - output file (.OUT), 24, 369
 - output variables, 199
 - PSpice A/D-equivalent parts, 174–175
 - simulation status window, 208, 368
 - starting, 206
 - using with other programs, 9
 - viewing in-progress output values, 368
 - waveform data file (.DAT), 14
- PSPICE.INI file, editing, 324

R

- real part, 361
- regulator, 95
- RELTOL (simulation option), 191
- resistors, 203, 363

S

- schematic page editor
 - starting other tools from
 - Model Editor, 101, 111–112
- scrolling, Probe windows, 346
- semiconductor
 - problems, 385
- shot noise, 245
- simulation
 - about, 2
 - analysis
 - execution order, 198
 - setup, 197
 - types, 196
 - batch jobs, 207
 - bias point, 374
 - failure to start, 58
 - initial conditions, 374, 378
 - output file (.OUT), 24
 - setup checklist, 56
 - starting, 206
 - status window, 208
 - troubleshooting checklist, 58
- simulation control parts, 60
 - ICn, 376
 - NODESETn (initial conditions), 376
 - PARAM, 67
- small-signal DC transfer analysis, 196, 225
 - introduction, 3
- Stimulus Editor, 33, 253
 - about, 9
 - creating new stimulus parts, 259
 - defining analog stimuli, 76
 - defining stimuli, 256
 - editing a stimulus, 260
 - manual stimulus configuration, 261
 - starting, 254
 - stimulus files, 253–254
- stimulus files, 11
 - configuring, 13, 120
- stimulus generation, 252

- manually configuring, 261
- stimulus, adding, 33
 - AC sweep, 233
 - DC sweep, 218
 - for multiple analysis types, 77
 - transient (analog/mixed-signal), 252
- subcircuits, 87
 - creating .SUBCKT definitions from designs, 91
 - creating .SUBCKT definitions from schematics, 115
 - tools to create, 91
 - ways to create/edit, 92
 - see also* models
- switch, 363
 - problems, 386
- system variables in expressions, 73

T

- table look-up parts (ABM), 153, 160, 174, 179
- temperature analysis, 196, 281
 - introduction, 6
 - with statistical analyses, 288
- TEMPLATE property, 140
 - and non-simulation parts, 140
 - examples, 143
 - naming conventions, 141
 - regular characters, 140
 - special characters, 142
- thermal noise, 245
- TIME (Probe output variable), 359
- total noise, 242
 - circuit, 245
 - per device, 245
- traces
 - adding, 28
 - direct manipulation, 344
 - displaying, 28, 35
 - markers, 334
 - output variables, 356
 - placing a cursor on, 30
- transformer
 - problems, 391
- transient analysis, 196
 - example, 32
 - Fourier analysis, 196
 - hysteresis curves, 266
 - internal time steps, 265
 - introduction, 5

- overview, 250
- problems, 388
- setting up, 34
- Stimulus Editor, 253
- stimulus generation, 252
- switching circuits, 266
- transient response, 263
- transistors, Darlington model, 95
- transmission lines, 362
- triode, 171
- troubleshooting
 - checklist, 58
 - missing DC path to ground, 84
 - missing ground, 83
 - unconfigured libraries and files, 81
 - unmodeled parts, 79
 - unmodeled pins, 82
- tutorials, *see* examples and tutorials

U

- unmodeled
 - parts, 79
 - pins, 82
- updating plots, 346

V

- VAC stimulus part, 75, 233
- variables in expressions, 73
- VDC (DC stimulus), 218
- VDC stimulus part, 74–75
- vendor-supplied parts, 61
- VEXP stimulus part, 75
- voltage comparator, 95
- voltage reference, 95
- voltage regulator, 95
- voltage source, controlled, 174, 192
- VPLOTn (write voltage plot part), 369
- VPRINTn (write voltage table part), 370
- VPULSE stimulus part, 75
- VPWL stimulus part, 75
- VPWL_F_N_TIMES stimulus part, 76
- VPWL_F_RE_FOREVER stimulus part, 75
- VPWL_N_TIMES stimulus part, 76
- VPWL_RE_FOREVER stimulus part, 75
- VSFFM stimulus part, 76
- VSIN stimulus part, 76
- VSRC stimulus part, 74–75, 78, 218, 233

VSTIM stimulus part, 33, 75–76

W

WATCH1 (view output variable part), 368

waveform analysis, 320

about, 8

adding traces, 28

cursors, 350

displaying simulation results, 28, 39

expressions, 364

functions, 364

hysteresis curves, 266

limiting waveform data file size, 334

multiple y-axes, 276

output variables, 356

for noise, 245, 363

performance analysis, 49, 274

placing a cursor on a trace, 30

plot, 321

printing Probe windows, 323

setting colors, 324

trace data tables, 349

traces, 334

traces, displaying, 344

traces, using output variables, 356

using markers, 331

waveform data file (.DAT), 14

waveform data file formats, 339

waveform families, 45, 221

waveform data file formats, 339

waveform families, displaying, 45

wavform analysis

arithmetic expressions, 364

worst-case analysis, 196, 306

collating functions, 287

example, 309

hints, 313

introduction, 7

model parameter values reports, 285

output control, 285

overview, 306

waveform reports, 286

with temperature analysis, 288

Z

zoom regions, Probe windows, 344

